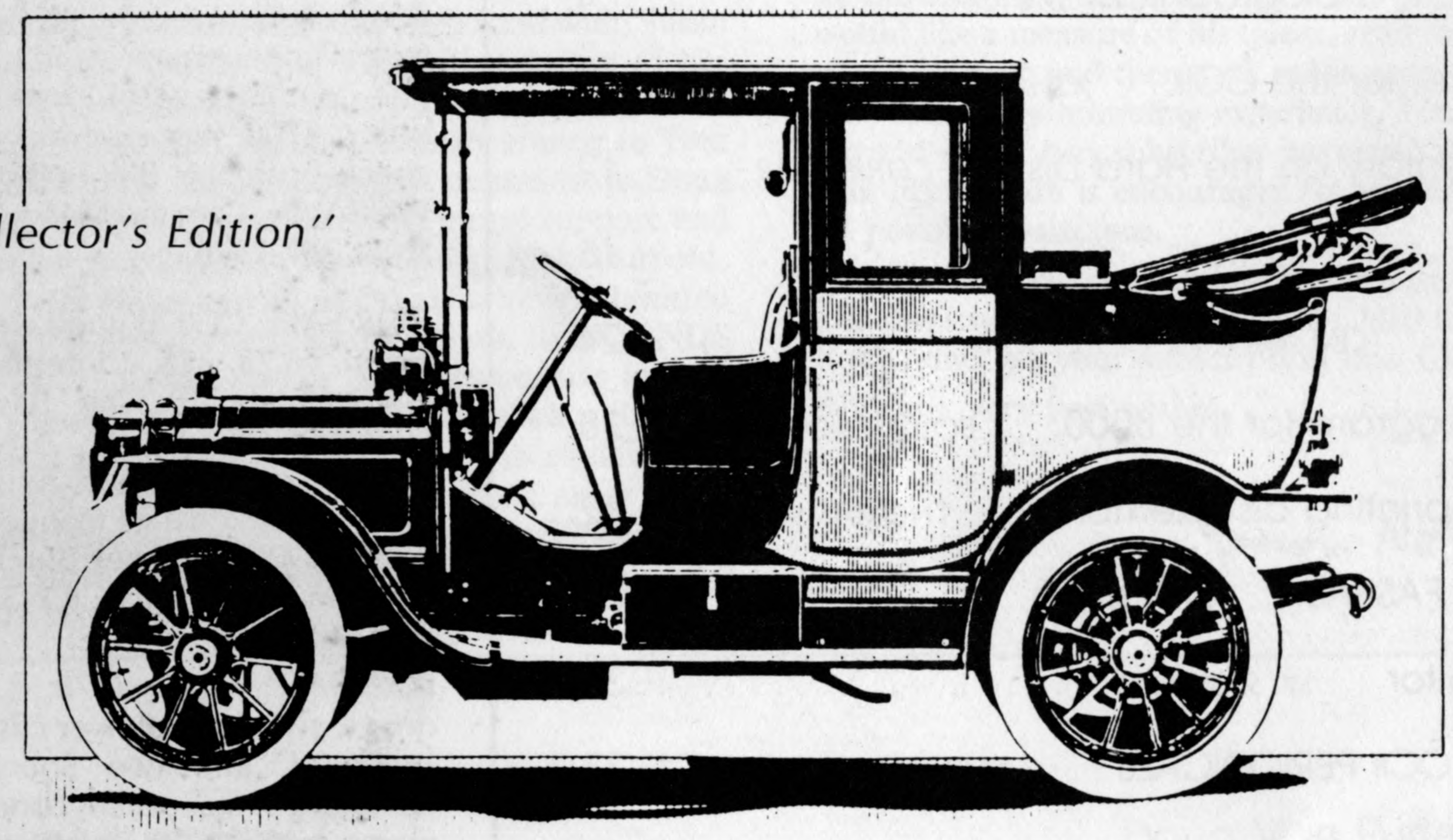
# COLORCUE

VOLUME VI
NUMBER 6

A BI-MONTHLY PUBLICATION BY AND FOR INTECOLOR AND COMPUCOLOR USERS

"Collector's Edition"



*from* **The Intecolor Corporation**

**The COMPUCOLOR II.**

with tributes by

***John Newby,*** *geologist*
***Doug Van Putte,*** *miracle worker*
***W. S. Whilly,*** *nuisance*
***Wallace Rust,*** *astronomer*
***Chris Zerr,*** *experimenter*
***Bob Mendelson,*** *8000 specialist*
***Tom Napier,*** *groom.*
***Thomas Wulff,*** *programmer*

# Colorcue

VOLUME VI, NUMBER 6 NOVEMBER/DECEMBER 1984

## CONTENTS

EDITOR: JOSEPH NORRIS COMPUSERVE: 71106, 1302

## UNCLASSIFIED ADS

FOR SALE: CCII, v6.78, 32K, deluxe keyboard, switchable lower case, handshake option, printer cable, sound, dust covers. Over 70 disks including Helm's Data Base, Screen Editor, Print; TermII, CC's Income Tax, plus many more. Many manuals and magazines. $1500 postpaid. Phil Simon, 6275 Cary Avenue, Cincinnati, OH 45224, 513-681-8370.

FOR SALE: CCII, v6.78, 32K, maintenance manual, all back issues of Colorcue, and the following software: Personal Data Base, Basic Language Tutorial 1-10, formatter, Basic editor, screen editor, assembler, Fortran, plus miscellaneous games and disk utilities. The computer is stil in good working order. $700, money order or certified check postpaid. Bill Anthony, 655 E. Wells Way, Camano Island, WA 98292. (206) 387-1576.

FOR SALE: CCII in good working condition, v6.78, 32K, 1 internal drive, extended keyboard. 'THE' Basic Editor in EPROM, Colorword, Tiny C, Tiny Pascal, and Forth. Lots of utilities and games, Maintenance manual, all Colorcue's. $500. Michael Burcham, 1707 Gleason, Iowa City, IO 52240. 319 354-2131.

FOR SALE: CCII, v6.78, 32K, 2 disk drives, switchable lower case, 101 key keyboard, and two Soundware attachments. In excellant condition. Programs include 10 Soundware games and 35 additional games, 20 applications including Equity, Bonds and Securities, Statistics I, text and Basic editors, Personal Data Base, Basic Tutor, assembler. Manuals include the Programming Manual, Maintenance Manual, Colorcue II-1 through II,6, Assembler Operating Manual. $1095, shipping prepaid. Mike Rousse, 15 South Owen Drive, Madison, WI 53705. (608) 238-1825, or leave message at (608) 233-6751.

COLORCUE is published bi-monthly. Subscription rates are US$12/year in the U.S., Canada, and Mexico (via First Class mail), and US$30 elsewhere (via Air Mail). All editorial and subscription correspondence should be addressed to COLORCUE, 19 West Second Street, Moorestown, NJ 08057, USA. (609-234-8117) All articles in COLORCUE are checked for accuracy to the best of our ability but cannot be guaranteed error free.

## from the Editor's Desk

*"Some assorted thoughts, thanks and numbers."*

I suppose the greatest reward for an editor is the privilege of producing an issue like this one. So many articles have been submitted, and of such high quality and excitement, that I feel a sense of having been renewed to meet the challenge. Several people have asked me at one time or another why I would want to take over a 'dying' publication. Well folks, I'm very suspicious of 'death' in the first place, and tend to see it as only transitional in the second. With this issue we pass our mantel to CHIP, and before starting my next article for that august publication I have some old business to complete.

Special thanks to the following: my friend and wife, Susan Hardee Norris for tutoring and supervision in use of her spectacular typesetter and showing me how to communicate successfully on the modem, and for her forbearance; to Tom Devlin for keeping me alive in desperate moments; to Doug Van Putte for endless encouragement, moral support and a steady stream of articles; to David Suits, Rick Taubold, Chris Zerr, Peter Hiner and all of you who have submitted materials to COLORCUE during my tenure; to FRIENDS JOURNAL, in Philadelphia for their generousity in providing the typesetting facilities; to all of you who, through your continued subscription, afforded me this enviable opportunity to know and work with you. I have made many special friendships through COLORCUE, and I have not know such a splendid collection of people as I have found among CCII enthusiasts.

My thanks, too, to my employer, The David Hafler Company, for the opportunity to gain some valuable computer experience on Intecolor products and for the motivation afforded by their utility of my computer output.

Some statistics: letters received, 387; letters written, 532; subscription revenues, $3800; publication costs and expenses, $5300; current subscribers, 205; pages in Volume VI, in excess of 200; subscriber cost per page, about 10 cents.

I hope you will expend the effort to assemble Tom Napier's program in this issue. It is simply spectacular, and an extraordinary example of first rate programming. If you would like a measure of his talent, read the article in Scientific American and then look at his code. You may find it, as did I, a very humbling experience. John Newby's gift is beyond value. Any subscriber interested in pursuing a hard disk installation is encouraged to contact John or me for all possible assistance.

Any materials submitted for publication that didn't make it in this issue will be forwarded to CHIP for their consideration....and get your subscription into CHIP!

With my very best wishes,

[illegible]

## "....*recursion*?"
## "What the dickens is '*recursion*'?"

Doug Van Putte
18 Cross Bow Drive
Rochester, NY 14624

When an object is defined by the application of a simpler case of itself, we have a recursive definition. Recursion can be useful in programming, as we shall see, but first let's explore the definition further. Consider a recursive definition of the factorial of the number 4. Ordinarily, we think of the factorial of 4 to be

```
4! = 4 * 3 * 2 * 1
```

But the factorial of 4 could also be expressed as

```
4! = 4 * 3!
```

which is a recursive definition, since 3! is used to define 4!.

Another distinctive feature of a recursive definition is that it must lead to a definite ending point. Let's expand the definition of 4! by continually repeating the above definition:

```
4! = 4 * 3!

3! = 3 * 2!

2! = 2 * 1!

1! = 1 * 0!

0! = 1
```

Our ending point is 0!, which is '1' by mathematical definition. Now let's compute the factorial by back-tracking from the ending point:

```
0! = 1

1! = 1 * 0! = 1 * 1 = 1

2! = 2 * 1! = 2 * ! = 2

3! = 3 * 2! = 3 * 2 = 6

4! = 4 * 3! = 4 * 6 = 24
```

We have computed our way back to the correct answer from the ending point using recursion.

The BASIC program in Listing I uses this same scheme to compute the factorial of a given number. After providing for the entry of a number whose factorial is to be computed, the main program stores this number in the first element of a 'stack', A(1). Then it successively breaks down this number by subtracting one from the number until zero is reached, each time storing the intermediate result in the next available location of the stack.

As long as the number entered is not zero (0! = 1), control is now sent to the subroutine. The subroutine will begin the factorial calculation by starting with the bottom element in the stack, A(I) = 1, and storing the factorial value of that element in FA. Note that unless there is only one element in the stack, the subroutine continues the calculation by calling itself recursively. Then, by the scheme shown above, when the first element is reached (our original number, at last!), the last factorial is computed and control is transferred back to the main program where the factorial of the number entered is printed.

```
00010 REM ***************** LISTING 1 ******************
00020
00030 REM ********RECURSIVE ROUTINE TO COMPUTE N!********
00040
00050 PLOT 12:DIM A(20):REM INCREASE DIMENSION FOR N>20
00060 INPUT "ENTER N ";N:NN=N
00070 IF NN=0 THEN FA=1:GOTO 160:REM SIMPLE CASE OF N=0
00080 I=0:REM INITIALIZE STACK INDEX
00090 REM TAKE APART N ONE BY ONE AND PUSH VALUES ON STACK
00100 I=I+1
00110 A(I)=N
00120 N=N-1
00130 IF N>0 THEN 100:REM DON'T STOP UNTIL N=0
00140 FA=1:GOSUB 200:REM 0!=1 IS WHERE FACTORIAL VALUE (FA)
00150 REM IS INITIALIZED
00160 PRINT:PRINT "FACTORIAL ";NN;" IS ";FA:PRINT
00170 GOTO 60:REM LETS GO AROUND AGAIN
00180 REM SUBROUTINE: POPS A VALUE FROM THE Ith LOCATION OF A
00190 REM STACK & COMPUTES ITS FACTORIAL BY USING A(I)*A(I-1)!
00200 IF I=0 THEN RETURN:REM ENDING POINT TEST
00210 FA=A(I)*FA:REM FACTORIAL ACCUMMULATION
00220 I=I-1:REM RESET STACK POINTER
00230 GOSUB 200:REM RECURSIVE CALL
00240 RETURN:REM END OF SUBROUTINE
```

## *PETER HINER:*

### A biographical sketch.

The editor has requested a brief biographical sketch of all the Colorcue authors. Peter Hiner has graciously responded and so we present this background on such an extraordinary talent as the author of a compiler must be.

Peter Hiner is 40 years old, married with two children, a boy of four and a girl of 18 months. He is employed by STC (Standard Telephones and Cables) as a System Design Manager in the Switching Division, which supplies telephone and data switching equipment, primarily to British Telecom. His engineering background is in logic and switching design for electronic switching systems. He claims his primary source of experience with software has been through the CCII which he bought in 1979. How did he get started?

"I followed what is probably the usual route of transcribing Basic games from magazines, to get used to our dialect of Basic, and then (had) a go at writing simple programs for games and graphics displays. From the beginning I had always believed that Assembly Language programming was the purest and most noble form of the art, so I soon started trying to break into this area. I am sure the first hurdles are the most difficult and the written materials now available for the beginner (particularly on input and output routines) must be of great help to those taking their first steps today. I tried the impossible task (for a beginner) of getting the required input and output routines from ROM, using a disassembler. This exercise left me totally lost and confused, although it probably started me on the path to writing a Basic compiler.

"I wrote a rudimentary sort of Invaders game—not very thrilling—and then got hooked on the idea of writing a ver-

The problem with making a recursive call in BASIC is that of preserving the values of the identifiers (e.g. the decreasing integer values in the above example.) This is done in our example by using a stack. The integer values are pushed onto the stack before the recursive call, and later popped off the stack in reverse order when executing the call. Some other languages, such as Pascal and 'C', can simplify this process considerably by 'keeping track' of the identifiers in a less cumbersome way.

"Can this recursion do anything useful?", you might ask. The answer is "You bet!". For example, consider the binary search program in Listing II. In this case we don't need a stack, just a scheme with a definite ending point. The scheme looks for a given value in an ascending array (it could be descending) and, if found, prints the index in the array where a matched element is located. A zero is printed for the index if a matching element is not found. The subroutine begins by comparing the middle element of the array with our value. If a match is not found, the array is split in two and only one half continues to be searched by the routine, recursively calling itself. Each time, then, the remaining part of the array is split until the middle element matches our value, or until the ending point is reached with no match. The ending point is reached when the lower bound of the array to be searched exceeds the upper bound. In either case (match or no match), control is returned to the main program to print out an index value. The value of recursion, here, is that it has been used to anchor a search method which is far more efficient than a simple linear search.

You might experiment with these principles by developing your own recursive routine in BASIC to raise a number to a power. I'll send you the answer to that one on request. I'd be interested in seeing any of your efforts.

```
00010 REM ***************** LISTING 2 ****************
00020 REM
00030 REM *********** BINARY SEARCH PROGRAM ***********
00040 REM
00050 PLOT 12:DIM A(21):BM=1:TP=21:REM TP IS NO. IN ARRAY
00060 FOR I=BM TO TP:REM READ IN ARRAY TO BE SEARCHED
00070 READ A(I)
00080 NEXT I
00090 PRINT:INPUT "ENTER ELEMENT FOR SEARCH ";X
00100 L=BM:H=TP:REM INITIALIZE SEARCH LIMITS
00110 GOSUB 130
00120 PRINT "ELEMENT INDEX IS ";B:GOTO 90
00130 IF L>H THEN B=0:RETURN:REM TEST FOR ENDING POINT
00140 M=INT ((L+H)/2):REM COMPUTE ARRAY MID POINT
00150 IF X=A(M) THEN B=M:RETURN:REM MATCH W/ MIDDLE ELEMENT?
00160 IF X<A(M) THEN H=M-1:GOTO 180:REM SEARCH BOTTOM 1/2 ARRAY
00170 L=M+1:REM SEARCH TOP 1/2 ARRAY
00180 GOSUB 130:REM RECURSIVE CALL
00190 RETURN:REM RETURN TO MAIN
00200 DATA 1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41
```

sion of the original adventure game to fit a 16K CCII machine without repeated disk reads. This was a magnum opus, using quite sophisticated techniques. The program consisted of a small interpreter (about 2K) driven by a 4K table of data. The huge amount of text (25K) for descriptions and messages was compressed into about 10K using a mixture of dictionary and other text compression techniques. This huge task took about a year to complete, but left me with a powerful tool for writing other adventures, and I was later able to produce a version of Scott Adam's Pirate Adventure in a few weeks.

"After that, I started on my Basic compiler which grew over a couple of years from humble origins to a complete implementation. By-products of this process were a tokenized form of Assembly Language to reduce file size, a pair of programs for disassembly and assembly, and a program for comparing files.

"I am approaching the end of the road as far as utility programs is concerned and am now turning my attention to artificial intelligence. I am interested in some simple forms of export system to assist in the process of reaching decisions based on a mixture of fact, experience and opinion. I don't expect that a Compucolor would support much more than a very simple system, but it should be an interesting field for experiment."

(The disassembler Peter refers to is available from the CHIP library. It permits entry of your own labels, and is designed to give every programming aid when disassembling totally foreign code. ED)

Peter includes in his note this challenging invitation: "One of the by-products of ZIP has been an Integer Interpreter with built-in debugging aids. It would be a relatively simple task to provide a similar disk-based version of normal Basic with built-in debugging aids, or anything else built in, if there is a requirement. The only limitation will be the reduced space available for user programs. If you have any ideas on this subject, let me know. I would intend to provide this Basic interpreter as a free program for the CHIP library."

# A HARD DISK FOR THE CCII

John Newby
4532 167th Avenue SE
Issaquah, WA 98027

For years I have operated with the limited storage capacity, the read/write errors, and speed variations of the Compucolor Disk (CD) drives. My CCII is used for many functions at work and I have an incredible amount of custom software, most of which uses large data files. An increasing disk problem has suggested a change of computer systems, even with the necessary conversion effort. I have always dreamed of a system with a Hard Disk (HD), but never figured one for the good old CCII.

In the March, April, and May 1983 issues of Byte, there appeared a series of articles by Cruce and Alexander on a hard disk interface for S-100 systems. It did not seem to be all that complex, with the exception of having to write an entirely new CPM type disk operating system. However, hard disk systems were going for nearly $2,000—much too much to sink into a CCII without even knowing if it would work. Even so, I began to consider the possibility.

In April 1984, a Seattle surplus electronics outlet, United Products Inc., advertised 5M byte hard disk drives for $250. I investigated and found they also had a nice controller for only $200 and a suitable power supply for $40. I had to try, so I purchased these two units.

For a week I pondered the construction of a disk operating system that would keep a map of all sectors and could randomly write to the disk as does CPM. This would eliminate the FCS problem of having to move all following files in order to delete a file. I could not stand to see the hard disk drive sitting around, so I decided to construct an interface adapter and write a short FCS compatible disk handler to see if I could get the thing working. I have never looked back.

Within two weeks, I put most of my floppy disks away and was having a great time. I discovered that FCS is not limited to 64K byte files. Since it is block structured, the actual file size limit is 64K sectors or 8.39M byte. Disk read/writes are so fast that file deletes are not much of a problem, particularly with a good file utility program. All FCS commands are operational, and any program that does not have its own disk routines will work with the HD. Other software, like Jim Helms' Data Base, requires some modification. However, the hard disk makes the data base much more effective. I regularly work with 250K to 300K byte data files without touching a floppy disk.

I have read horror stories about disk crashes, and it would take over 100 CD floppy disks to backup the hard disk. I thought about a tape drive backup system, but it is fairly expensive. I made another trip to United Products. This time they had just received in stock remanufactured 10M byte drives and were selling them for $275. Hmmm. The controller and power supply on the 5M byte drive will handle two drives, so why not use the second for backup? Done! I am now operating a 10M byte drive (actually 11.90M bytes) and using the 5M byte drive (actually 5.95M bytes) as a backup. It takes about 12 minutes to fill the backup drive. The hard disk system works very well and I do not think I would again use a CCII without a hard disk system.

Here are some of the features of the hard disk system:

1. The FCS HD handler is located in the open ROM space (4000-5FFFH), and only occupies 805 bytes. You will need to have either a single rom board or Freepost's bank select ROM board. The FCS system ROM also requires slight modification.

2. The disk drive is logically configured as eight 1.49M byte devices, HD0: to HD7: (four for a 5M byte drive).

3. Each device directory defaults to a 32 sector size (the maximum allowable in FCS); however, there is room for 191 files on each logical device.

4. There is no limit to file size since FCS can access up to 64K block (8.39M byte) files.

5. All FCS commands are operational, including copying between CD and HD devices.

6. The system is fast. As a benchmark, a 32K byte write on the HD takes 2.0 seconds compared to 30 seconds for the CD. A 32K byte read is 1.5 seconds compared to 13 seconds. When you load screens, they 'pop' right up. For disk intensive operations, such as loading LDA files, data base sorts, or Fortran compiling and linking, the speed increase is greater.

This article describes the major elements of a hard disk system consisting of the following items: a custom CCII SASI Hard Disk Interface Adapter, a hard disk controller, one 10M byte or 5M byte hard disk drive, a power supply, enclosure, and cable, and changes to the FCS operating system. I suggest you also read the series of articles by Cruce and Alexander.

You will need to write a utility program to format and check the drives. If you use another drive as backup, you will need to write an additional utility routine. This is not too difficult, and all of the subroutines required are in the HD handler presented here. If you wish, I will send you the source for my utility routines.

As an aside, I also built an EPROM programmer which plugs into the CCII fifty pin bus and will provide the circuit diagram and software required to anyone who asks. It makes life as a CCII hacker much simpler.

## THE SASI INTERFACE

The Shugart Associates Standard Interface (SASI) is an industry standard interface for parallel data and control signal transfers between a host computer and a hard disk controller. The circuit diagram for the CCII SASI interface

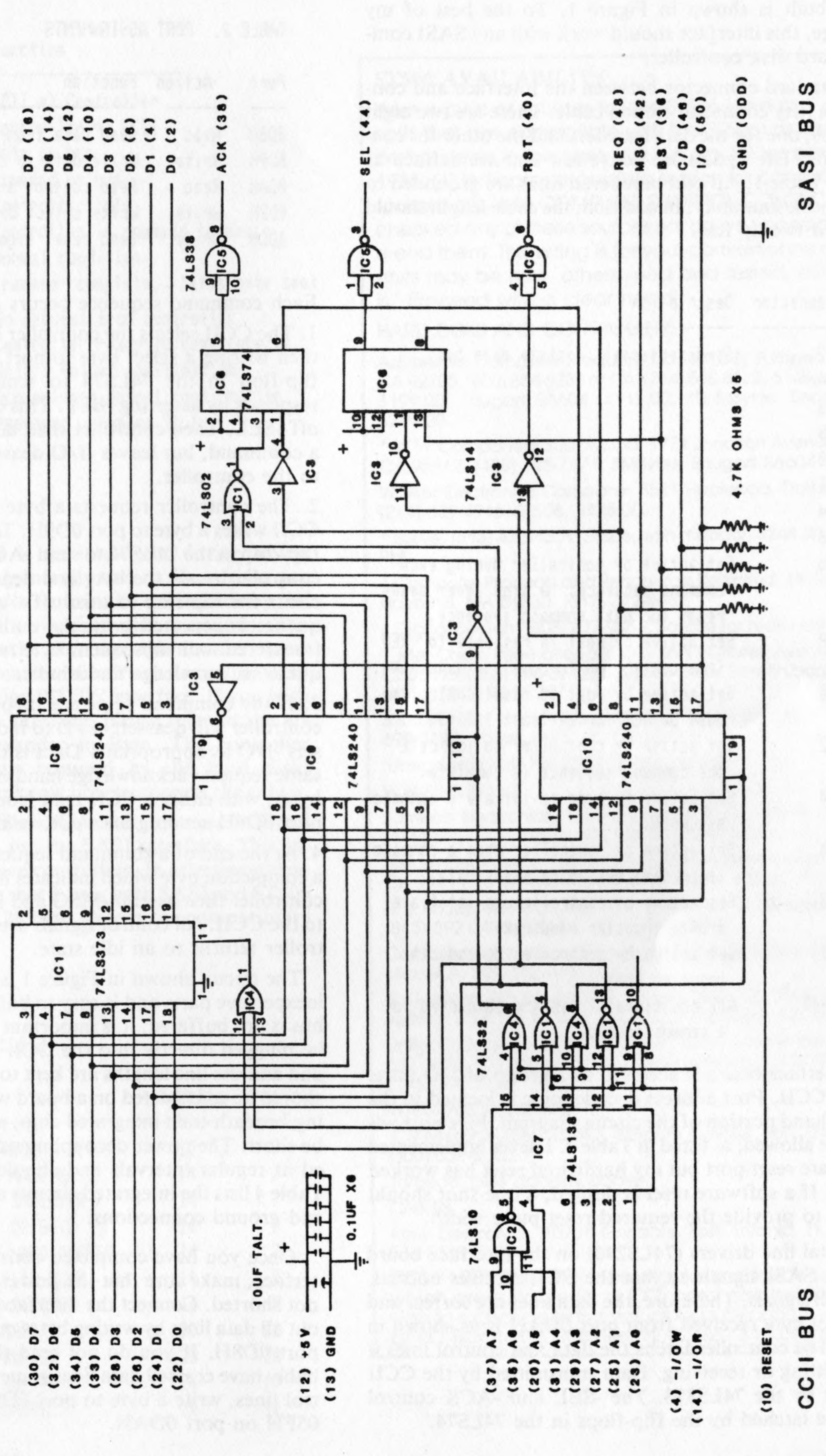


Figure 1. Compucolor II SASI Hard Disk Interface Adapter

board I built is shown in Figure 1. To the best of my knowledge, this interface should work with any SASI compatible hard disk controller.

The standard connector between the interface and controller is a fifty connector ribbon cable. There are two eight line groups, one for the data transfers and the other for control signals. The signals are active low and are defined as shown in Table 1. All odd numbered lines are grounded to improve noise immunity. In addition, the cable length should not exceed twenty feet.

| Signal | Connector | Description |
|---|---|---|
| -DB0 | 2 | Bi-directional tristate data bus. |
| -DB1 | 4 | |
| -DB2 | 6 | |
| -DB3 | 8 | |
| -DB4 | 10 | |
| -DB5 | 12 | |
| -DB6 | 14 | |
| -DB7 | 16 | |
| -BSY | 36 | Set active by controller during each command sequence. A high level means ready for next command sequence. |
| -ACK | 38 | Set active by host in response to -REQ from controller to complete handshake. |
| -RES | 40 | Set active by host to reset controller. Must be low for at least 100 nS. |
| -MSG | 42 | Set active by controller to indicate the command sequence is complete. |
| -SEL | 44 | Set active by host to intiate a command sequence. |
| -C/D | 46 | Set active by controller when a command is on data bus. High means data. |
| -REQ | 48 | Set active by controller to intitiate a byte transfer handshake. |
| -I/O | 50 | Set active by controller to indicate input to host. |
| GND | 1-49 | All odd signal lines are connected to a common ground. |

The interface board is accessed as a group of I/O ports from the CCII. Port address decode logic is located in the lower lefthand portion of the circuit diagram. Five port addresses are allowed, as listed in Table 2. I never implemented the software reset port but my hardwired reset has worked very well. If a software reset is desired, a one-shot should be added to provide the required reset pulse width.

The octal line drivers (74LS240) on the interface board invert the SASI signals so that the CCII receives normal, active high signals. Therefore, the data lines are correct and the control byte received from port 0DAH is as shown in Table 3. The controller latchs the data and control lines it is transmitting or receiving. Data transmitted by the CCII is latched by the 74LS373. The -SEL and -ACK control signals are latched by the flip-flops in the 74LS74.

TABLE 2. PORT ASSIGNMENTS

| Port | Action | Function |
|---|---|---|
| 0D8H | Read | Read data from controller |
| 0D9H | Write | Write data to controller |
| 0DAH | Read | Read control signals |
| 0DBH | Write | Write select byte to controller |
| 0DCH | Write | Send reset signal (not implemented) |

Each command sequence occurs as follows:

1. The CCII selects the controller by waiting until it is idle, then writing a select byte to port 0DAH. The write sets a flip-flop in the 74LS74 to send -SEL. The controller responds by asserting -BSY. This clears the flipflop to turn off -SEL. Then controller then activates -C/D to indicate a command, but leaves -I/O deasserted to indicate output to the controller.

2. The controller requests a byte by asserting -REQ. The CCII writes a byte to port 0D9H. This write also sets another flip-flop in the 74LS74 to send -ACK to the controller. The controller reads the byte and deasserts -REQ, which also clears the flip-flop to turn off -ACK. This handshake sequence is repeated until the entire six byte command is transfered. All subsequent byte transfers use the same request/ acknowledge handshake.

3. If the command sequence involves a data transfer, the controller will deassert -C/D to indicate a data transfer and sets -I/O as appropriate. Data is then transfered using the same request/ acknowledge handshake as for the command bytes, with either a CCII read from port 0D8H or write to port 0D9H sending an -ACK to the controller.

4. At the end of a command sequence, the controller sends a completion byte which indicates if any errors occured. The controller then asserts -MSG and a message byte (0)is sent to the CCII. All control signals are deasserted and the controller returns to an idle state.

The circuit shown in Figure 1 is comprised of relatively inexpensive parts and is easy to build. As the CCII fifty pin bus is not buffered, it is important that the interface board be plugged directly into the CCII and that the CCII data and address line lengths are kept to a minimum. The circuit should be constructed on a board with a ground plane passing beneath each integrated chip, and ground leads should be short. The power decoupling capacitors should be placed at regular intervals and should also have short leads. Table 4 lists the integrated circuits and indicates their power and ground connections.

Once you have completed construction of the SASI interface, make sure that the power lines (or any others) are not shorted. Connect the interface to your CCII and check out all data lines by writing bytes out port 0D9H and reading port 0D8H. If you do not read the same value, you probably have crossed data lines somewhere. To check the control lines, write a byte to port 0D9H and you should read 05FH on port 0DAH.

Table 3. CONTROL BYTE

| Bit | Signal | Level | Function |
|---|---|---|---|
| 0 | I/O | 0 | CCII => Controller |
| | | 1 | Controller => CCII |
| 1 | C/D | 0 | Data on bus |
| | | 1 | Command on bus |
| 2 | BSY | 0 | Controller idle |
| | | 1 | Controller in command sequence |
| 3 | MSG | 0 | Normal condition |
| | | 1 | Transfer complete, status byte sent |
| 4 | REQ | 0 | No request from controller |
| | | 1 | Controller ready to send/receive |
| 5 | SX | 1 | Select output confirmation |
| 6 | AX | 1 | Acknowledge output confirmation |
| 7 | RES | X | Reserved - Forced to 0 |

## THE CONTROLLER

The hard disk controller is an intelligent device which operates hard disk drives (generally two to four drives per controller). The controller is sent simple commands by the computer and performs the required disk functions. The controller is connected to the disk through another interface. Most five and one-quarter inch drives use an industry standard Seagate Technology 'ST506' interface. If you wish to know more about this interface, please refer to the articles by Cruce and Alexander in Byte. The controller handles all of the signals required by the hard disk; therefore, you only have to know how to connect the cables.

Most hard disk controllers which do not use a direct memory access interface, use the SASI interface. This includes almost all inexpensive controllers. In selecting a controller, you should be sure that it uses both SASI and ST506 interfaces and supports 128 byte sectors for compatibility with FCS.

TABLE 4. INTEGRATED CURCUITS

| Number | Type | + 5 Volts | Ground |
|---|---|---|---|
| IC1 | 74LS02 2-INPUT NOR GATE | 14 | 7 |
| IC2 | 74LS10 3-INPUT NAND GATE | 14 | 7 |
| IC3 | 74LS14 SCHMITT INVERTER | 14 | 7 |
| IC4 | 74LS32 2-INPUT OR GATE | 14 | 7 |
| IC5 | 74LS38 2-INPUT NAND BUFFER | 14 | 7 |
| IC6 | 74LS74 DUAL D FLIP-FLOP | 14 | 7 |
| IC7 | 74LS138 1-OF-8 DECODER | 16 | 8 |
| IC8 | 74LS240 OCTAL BUFFER | 20 | 10 |
| IC9 | 74LS240 OCTAL BUFFER | 20 | 10 |
| IC10 | 74LS240 OCTAL BUFFER | 20 | 10 |
| IC11 | 74LS373 OCTAL LATCH | 20 | 10 |

## ST506 AVAILABILITY.

Here is a list of sources for the ST506 Hard disk drive (and units that are compatible with the ST506) as well as other components in the hard disk system. Prices are from May 1985. [*] indicates a possible source for controllers. These sources are from COMPUTER SHOPPER. COLORCUE has not checked any of these sources nor do we necessarily recommend them. This listing is for your convenience only. Some units may be new, others used and tested, still others 'as is.' Proceed with a clear head!

### HARD DISKS AND CONTROLLERS.

Advanced Computer Products, Inc. 1310 E. Edinger, Santa Ana, CA 92705. (800) 854-8230 or CA: (714) 558-8822. 6 MBytes: CHST506, $199.00; Shugart SA604, $119.00. 10 MBytes: Seagate ST419, $299.00.

* W W Component Supply, Inc. 1771 Junction Avenue, San Jose, CA 95112. (408) 295-7171. 5MBytes: Shugart SA604, $ 149.00.

Walker Electronics Company. 3521 Hacienda, Dallas, TX 75233. (214) 339-4916. ST506, $175.00.

* Steve, (513) 433-1501. ST506, with manual, $249.00. Call after 6 PM.

* Computer Products and Peripherals Unlimited. 18 Granite Street, Haverhill, MA 01830. (617) 372-8637.

* Digital Search. (803) 877-9444. Source for many drive products.

* Met-Chem International Corp. 2911 Dixwell Avenue, Hamden, CT 06518. (203) 248-3212, (800) 638-2436. Bulletin board (300/1200 baud) (203) 281-7287.

John Hanson. 1110 Pheasant Circle, Winter Springs, FL 32708. (305) 699-0124. ST506, $195.00.

(Unsigned) (916) 726-3291. ST506, $95.00.

### POWER SUPPLIES

Jameco Electronics, 1355 Shoreway Road, Belmont, CA 94002. (415) 592-8097. Kepro TDK $59.95 (12 volts at 2 A)

Nicorn Electronics. 10010 Canoga Avenue, Unit B-8, Chatsworth, CA 91311. (8180 341-8833. Apple Power Supply (12 volts at 2.5A)

H. J. Knapp of Florida, Inc. 4750 96th Street, St. Petersburg, FL 33708. (813) 392-0406. $29.95. (12 volts at 3.5A)

Computer Products and Peripherals Unlimited. (See address above) Model A. $29. (12 volts at 3A)

B. G. Micro. PO Box 280298, Dallas, TX 75228. (214) 271-5546. (12 volts at 2.8A, 12 volts at 2A) Recommended. $37.50.

United Products, Inc. 1123 Valley Street, Seattle WA 98109. (206) 682-5025. TDK Model 21145. Recommended. $34.50. (12 volts at 2.8A and 2.0A)

### INTEGRATED CIRCUITS

JDR Microdevices. 1224 S. Bascom Avenue, San Jose, CA 95128. (800) 538-5000.

Jameco. (See address above.)

### CABLES

Altex Electronics. 10731 Golfdale, San Antonio, TX 78216. (800) 531-5369.

COMPUTER SHOPPER also regularly advertises enclosures which are suitable for a CCII hard disk installation. If you wish to subscribe to this valuable publication, write to them at PO Box 1419, Titusville, FL 32781. $15.00/year.

TABLE 5. OMTI 20L COMMAND SUMMARY

| Code | Command | Explaination |
|---|---|---|
| 00H | SENSE STATUS | Check for drive ready |
| 01H | RECALIBRATE | Step out until Track 0 |
| 03H | REQUEST SENSE | Report detailed error codes |
| 04H | FORMAT DRIVE | Format entire drive |
| 05H | CHECK TRACK | Check entire track for errors |
| 06H | FORMAT TRACK | Format a single track |
| 07H | FORMAT BAD TRACK | Write defective bit in ID field |
| 08H | READ DATA | Read 1 to 256 sectors |
| 0AH | WRITE DATA | Write 1 to 256 sectors |
| 0EH | ASSIGN ALT TRACK | Set alternate track bit in ID field |
| C2H | ASSIGN PARAMETERS | Assign hard disk variable parameters |
| E1H | WRITE ECC | To allow ECC testing |
| E2H | READ ID | To read ID field |
| E6H | REQUEST LOGOUT | Read retry and error counts |
| EAH | READ ECC | To allow ECC testing |
| ECH | READ DATA BUFFER | Read buffer only, not disk |
| EFH | WRITE DATA BUFFER | Write buffer only, not disk |

I used an OMTI 20L controller. Its general characteristics include:

1. A 10K byte buffer to allow reading and writing of a full track at one time by the controller.

2. Controller logic allows 128, 256, or 512 byte sectors.

3. A single ID field is used for each track, allowing for thirty-eight 256 byte sectors (seventy-six 128 byte equivalents), as opposed to standard formatting which results in only 32 sectors. Therefore, the capacity is 5.95M bytes on a 5M byte drive and 11.90M bytes on a 10M byte drive.

4. Four bytes of error correcting code (ECC) is written for each 256 bytes. This allows automatic correction of up to five bits per 256 bytes during disk reads.

5. Data written to the disk can be automatically verified.

6. Up to 256 sectors can be transfered in a single read or write command.

7. Device size limited to 2 to the 15th power sectors (536M bytes for 128 byte sectors) or 65K tracks (637M bytes).

8. The controller is on a single 5.75 by 8.00 inch PC board which mounts directly on the hard disk drive. (I had to place a copper PC board ground plane between the controller and hard disk circuit boards to eliminate interference.)

Table 5 summarizes the SASI commands supported by the OMTI 20L controller. If bad sectors are found on any track, alternate tracks can be assigned during formatting; from then on, access to these tracks is transparent to the host computer. However, I have not encountered a bad track in three hard disk drives, two of which were remanufactured.

The hard disk handler code described in this article should work with minor changes, if any, on most any SASI compatible controller which supports 128 byte sectors. The handler only uses the normal Sense Status, Request Sense, Read Data, Write Data, and Assign Parameters commands. A separate utility program must be written to format and check the drive (and assign alternate tracks, if necessary).

OMTI no longer produces the 20L controller; however, there may still be some on the surplus market (United Products is sold out). OMTI's current comparable controller is the 5200 series. This controller has the same, or better, features and supports 128 byte sectors, but uses the standard thiry-two 256 byte sectors per track. However, it also supports eight inch disk drives. OMTI is now part of Scientific Microsystems, and their products are distributed by Arrow Electronics.

The industry standard controller is the Data Technology Corporation DTC-500 Series. These are carried by Active Electronics and others. However, I do not know if these controllers support 128 byte sectors.

## THE HARD DISK DRIVE

The hard disk drive is comprised of several sealed metal media disks rotating at 3600 rpm. There is a read/write head for each surface (two per disk). The ST506 drive has two disks and four heads. Each surface has 153 tracks for a total of 612. The ST412 also has four heads; however, later technology allowed for 306 tracks per surface, doubling the capacity. The ST412 also has a much faster head stepping speed.

About the only requirement for hard disk selection is that it should be 'ST506 compatible,' and most five and one-quarter inch hard disk drives are. The controller must be assigned the proper parameters using the Assign Parameters command. Variable parameters you will need to know include those shown in Table 6.

Seagate Technology hard disk drives have a 16 pin (8 connection) option shunt block which must be inserted to set customer options. This includes the drive select encoding. For ST506 and ST412 drives, you should shunt pins 2-15, 4-13, and 8-9 (DS1) or 7-10 (DS2). Your drive supplier should provide you with a description of how to set the options for your drive.

## THE POWER SUPPLY, ENCLOSURE, AND CABLES

Most hard disks require power supplies which can provide +12 volts at 3 to 4 amps peak (2 amps continuous) and +5 volts at 1 amp. The peak 12 volt current is only required while the drive is coming up to speed. I used a Kepro switching power supply which is generally available on the

TABLE 6. DRIVE PARAMETERS WITH OMTI 20L CONTROLLER

| Name | ST506 | ST412 |
|---|---|---|
| Step Pulse Width | 3 uS | 2 uS |
| Step Pulse Period | 3 mS | 50 uS |
| Step Mode | normal | normal |
| Number of Heads | 4 | 4 |
| Total Number of Tracks | 612 | 1224 |
| Reduced Write Current Track | 128 | 128 |
| Sectors per Track | 76 | 76 |

TABLE 7. PARTS COST FOR HARD DISK SYSTEM

| | |
|---|---|
| SASI Interface Board and Parts | $ 50.00 |
| All Cables and Connectors | 75.00 |
| Kepro Power Supply | 35.00 |
| Heathkit Enclosure Parts | 90.00 |
| Fan | 25.00 |
| OMTI Controller | 200.00 |
| Seagate Technology ST412 Drive | 275.00 |
| Subtotal | 750.00 |
| ST506 Backup Drive | 250.00 |
| Total | 1000.00 |

surplus market for less than $50. As long as I only power up one drive at a time, this supply runs a ST506 drive, a ST412 drive, and the OMTI controller.

I have used Heathkit H-77 disk drive enclosures for both my two CD drives and my two HD drives. The cabinets look nice and have plenty of room for two drives and a power supply. You can order the eleven pieces required from Heathkit for about $90. [See parts list in Table 8.] You should also put a fan on your enclosure to provide plenty of ventilation. Hot components are not known for their reliability or long life.

Connection cables are a significant item in putting togeather a hard disk system. You will need the following:

1. A fifty line ribbon cable from the interface to the controller, along with edge or pin connectors.

2. A thirty-four line ribbon cable from the controller to each drive in a daisy chain manner, along with connectors.

3. An individual twenty line ribbon cable from the controller to each drive, along with connectors.

4. A four wire connection from the power supply to each drive and to the controller. The connector is an AMP 1-480424-0.

## THE OPERATING SYSTEM CHANGES

All FCS routines access storage devices by setting up a series of parameters and then calling a handler for the current device type. There is a table in the FCS ROM which lists the power up default device type and number. This is followed by a jump to the handler, a device name, and the maximum device number for each supported device. To support the 'HD' drives, the disk lookup table in the FCS rom must be modified as shown in Listing 1.

The hard disk handler can be located anywhere in the 4000-5FFFH ROM space. In order to support Freepost's multiple bank ROM board, the jump to the handler in FCS goes to a patch (Listing 2) at the end of each ROM bank. This patch jumps to the handler located in Bank 0. In this manner, the hard disk can be accessed from any bank with the patch at the end.

NOTE: There is a routine called 'RESET' which is called on entry to FCS. RESET sends a turn off function to each device in the device table. If you are in a bank (including the RAM bank) that does not have the patch in place, your system will hangup. For the RAM board, you can CPU reset; ESC W to Basic Reset; set the bank by OUT 255,7; defeat calls to the hard disk handler by POKE 24542,201; and then ESC D to FCS.

My hard disk handler code is presented as Listing 3. It is for a Seagate Technology ST412 disk drive and an OMTI 20L controller. The modifications for a ST506 drive are also indicated. If you use another drive, the code will need to be appropriately modified. A good manual on the controller you select is a must. The handler occupies only 805 bytes, leaving plenty of room for your other utilities or programs.

There is nothing magic about the logical device size I selected. You can easily modify the handler and device table to support other numbers of logical devices. The number of sectors per device should be an even division of the total number of sectors on the hard disk, and must not exceed 64K sectors (FCS's limit).

The only limitation with the handler, as I have implemented it, is that you cannot read or write data directly from a RAM card occupying the 4000-5FFFH memory area, unless you load a copy of the handler in the RAM bank. I get around this by either using a CD disk or by using a simple utility program which loads the data in upper memory from the hard disk, switches to the ram bank, and then moves the data down.

## CONCLUDING REMARKS

The total cost to setup my hard disk system is about $1000, and can be broken down approximately as shown in Table 7. I have recently seen ST506 drives on the surplus market for less than $175 and new Shugart 604 6.7M byte drives advertised for $139.

When you sit back and consider having eight 1.5M drives on line, very fast disk access, essentially unlimited file size, full FCS compatibility, and not having to mess with CD disk drives and limited capacity disks, it all seems well worth the effort and price. My CCII is now so powerful and so much fun to use I will probably stick with it for several more years. Good old Serial No.16 is a lot different than when it rolled off the assembly line in 1978. When I eventually move on, the entire hard disk system (beyond the interface board), which is industry standard, and can move on with me.

References.

Andrew C. Cruce and Scott A. Alexander. "Building a Hard-Disk Interface for an S-100 Bus System", a three part article in Byte: March, April, and May 1983.

Jim Thoreson. "The Winchester Odyssey, from manufacturer to user." Byte: March 1983.

"ST506 OEM Manual." July 4, 1983. Seagate Technology, 920 Disk Drive, Scotts Valley, California 95066. 408/438-6550

Scientific Microsystems (OMTI). 339 North Bernard, Mt. View, California 94043, 415/964-5700.

United Products Inc. 1123 Valley, Seattle, Washington 98109 206/682-5025.

# *Commentary on the* **HARD DISK**

Chris Zerr
10932-156th Court
Redmond, WA 98052

Wouldn't you know it! Just as COLORCUE ends, something new appears for the CCII to be shared with all users—a hard disk system. When John Newby called me, back in May of 1984, and told me about this I fell off my chair! Amazing! So last February I splurged and bought a ST506 Hard Disk, OMTI 20L controller and a power supply. The total cost was $506 [1], including cables and wire. I began to wire the SASI interface using a Radio Shack epoxy board and point to point wiring.

It took me a week to wire, working a few hours every night. Finally the big day arrived. I plugged it in and...hmmm. I had a broken wire and one wire misplaced. I corrected these and..Bingo..it worked. But this was only the interface being tested. The next step was to connect the interface to the controller and hard disk. This is so simple that there isn't much that can go wrong, right? All that remained was connecting two jumpers, wiring a select strap, and connecting a cable between the CCII interface and the controller card.

I turned on the power and typed DIR HD0: . FCS ERROR ENVE. Disk not initialized..whew! I next had to format the hard disk and initialize each directory. The formatting takes only thirty seconds or so, and initializing is nearly instant. Once completed, the experience is wonderful. To think, 6 megabytes online. Editing SRC files is a breeze. I do not have a second drive for backup so, for now, I only backup files I have changed or that are really important. As for software, I am currently using the Frepost bank board with the HD driver in Bank 0. So far I'm enjoying all of it. The CCII will keep me going for a few more years now. Should it die, I'll know that the hard disk can move on to another computer. Only the interface card, which costs about $50.00 will be lost.

I would like to give special thanks to John Newby for his time and effort in making this wonderful enhancement possible.

[1] The cost could also be as low as this for subscribers who want to investigate a hard disk for their CCII. See the list of possible sources for the components in this issue. ROM chips are available from John Newby. Joseph Norris might lay out a double sided PC card for the interface. You may place an order with him. The price is not known, but is expected to be about $40.00.

```
;       LISTING 1.  FCS ROM MODIFICATION
;
        CDHD    EQU     Ø211CH  ;(1AC1)
        HDHD    EQU     Ø5FDEH  ;(5FDE)
;
        ORG     Ø368BH          ;(ØØ4Ø)
;
;       INITIAL DEVICE NAME AND NUMBER
;
        IDEV:   DB      'CD'
        IUNIT:  DB      'Ø'
;
;       COMPUCOLOR DISK HANDLER
;
        HDVCT:  JMP     CDHD    ;VECTOR TO HANDLER
        CDNM:   DB      'CD'    ;NAME
        CDNU:   DB      2       ;MAX. DEVICES
        CDSEC:  DB      ØØAH    ;SECTORS/TRACK
        CDK:    DB      Ø5ØH    ;UP TO SPEED TIME
                DS      2       ;SPARE SPACE
;
;       INSERT HARD DISK HANDLER HERE
;
        HDHDL:  JMP     HDHD    ;VECTOR TO PATCH AT END OF ROM
        HDNM:   DB      'HD'    ;NAME
        HDNU:   DB      8       ;4 FOR ST5Ø6
                DS      4
;
;       ROOM FOR STILL ONE MORE
;
        OPNHD:  DB      ØFFH
                DS      5
```

```
;       LISTING 2.  BANK ROM PATCH
;
        BFLAG   EQU     Ø8Ø42H  ;SPACE FOR CURRENT BANK
        BANK    EQU     Ø??H    ;THE CURRENT BANK
        HDHAND  EQU     Ø????H  ;WHEREVER YOU WANT IN 4ØØØ-5FFFH
;
        ORG     Ø5FDEH
;
        HDHD:   MVI     A,BANK  ;THIS ROM BANK
                STA     BFLAG   ;SAVE IT FOR RETURN
                XRA     A       ;GOTO HANDLER BANK
                OUT     ØFFH
                CALL    HDHAND  ;DO THE HANDLER OPERATION
                PUSH    PSW     ;SAVE STATUS
                LDA     BFLAG   ;GO BACK TO PROPER BANK
                OUT     ØFFH
                POP     PSW     ;RESTORE STATUS
                RET
;
;       OTHER BANK ACCESS ROUTINES
;
        RRET:   POP     B       ;GET BACK BANKS
                POP     H       ;GET ORIG. CALLERS RETURN
                MOV     A,B     ;GO TO PROPER BANK
                JMP     RJMP
;
;       RCALL - USED TO CALL A ROUTINE IN ANOTHER BANK
;               HL = ROUTINE ADDRESS
;               B  = CALLING BANK
;               C  = ROUTINE BANK
;               NOTE: ROUTINE CAN NOT ALTER STACK
;
        RCALL:  PUSH    B       ;SAVE BANKS
                LXI     D,RRET  ;SET UP RETURN
                PUSH    D
                MOV     A,C     ;GET READY TO GO
;
;       THE NORMAL MINIMUM PATCH REQUIRED
;
        RJMP:   OUT     ØFFH    ;GO TO BANK
                PCHL            ;SHOULD BE AT Ø5FFFH
```

```
;       LISTING 3.
;
;       COMPUCOLOR II - SEAGATE TECHNOLOGY/ OMTI 20L DISK HANDLER
;       FCS ROM VERSION 6.78 (8.79)
;       VERSION 1.0 : 5/1/84 : ST506 DRIVE, 4 * 1.49M BYTES
;               2.0 : 9/1/84 : ST412 DRIVE, 8 * 1.49M BYTES, TRAILERS
;               2.1 : 4/1/85 : FIXED POWER UP ERROR, INTERRUPT MASK
;
;       SYSTEM ROUTINES  6.78    (8.79) *****************************
;
        CRLF    EQU     0338BH  ;(17C1) CR AND LF TO SCREEN
        LBYT    EQU     0339BH  ;(17D1) LIST HEX BYTE TO SCREEN
        OSTR    EQU     033F4H  ;(182A) SEND STRING TO SCREEN
        MOVDH   EQU     0343BH  ;(1871) MOVE B BYTES FROM HL TO DE
        CMPHD   EQU     0344DH  ;(1883) COMPARE DE TO HL
        SUBHD   EQU     03459H  ;(188F) SUBTRACT DE FROM HL
        PSPAC   EQU     034B3H  ;(18E9) PRINT A SPACE
        PCOLN   EQU     034B8H  ;(18EE) PRINT A COLON
        PSSTR   EQU     034BDH  ;(18F3) PRINT A SPACE AND STRING
        PSTR    EQU     034C0H  ;(18F6) PRINT A STRING
        BC2BK   EQU     035A8H  ;(19DE) CONVERT BYTE COUNT TO BLOCKS
        ERMSG   EQU     03631H  ;(1A67) ERROR MESSAGE
        HDNAM   EQU     0369BH  ;(0050) 'HD'
;
;       DATA AREAS *************************************************
;
;       AREAS OTHERWISE USED BY CDHD
;
        SBC     EQU     08042H
        CRC1    EQU     08043H
        CRC2    EQU     08044H
        BRTRY   EQU     080E0H
        RFLG    EQU     080E1H
        CRTRY   EQU     080E2H
        OSEC    EQU     080EDH
        SEC     EQU     080EEH
        TRK     EQU     080EFH
        TEMP2   EQU     081F4H
;
;       AS USED BY HD HANDLER
;
        BFLAG   EQU     08042H  ;CURRENT OPERATING BANK NUMBER
        HALF1   EQU     08043H  ;NO. SECTORS TO READ/WRITE
        HALF2   EQU     08044H  ;BALANCE OF SAME IF OVER 256
        HDFLG   EQU     08043H  ;MORE THAN ONE ERROR FLAG
        SECT0   EQU     080E0H  ;SECTOR ADDRESS MSB'S
        SECT1   EQU     080E1H  ;SECTOR ADDRESS
        SECT2   EQU     080E2H  ;SECTOR ADDRESS LSB'S
        LBLK    EQU     080EDH  ;LAST BLOCK BYTE COUNT
        BKCNT   EQU     080EEH  ;BLOCK COUNT FOR TRANSFER
        ECODE   EQU     080EDH  ;ERROR CODE
        RSFLG   EQU     081F4H  ;0 = CPU RESET PERFORMED
;
;       THESE ARE THE SAME FOR ALL HANDLERS
;
        TFCN    EQU     080E5H  ;FUNCTION CODE
        TDRV    EQU     080E6H  ;DRIVE NUMBER
        TBLK    EQU     080E7H  ;BLOCK NO FOR TRANSFER
        TMEM    EQU     080E9H  ;MEMORY BUFFER POINTER
        TBC     EQU     080EBH  ;BYTE COUNT
;
;       ST412 DISK PARAMETERS    (ST506) ****************************
;
        PLSEW   EQU     002H    ;(0003) STEP PULSE WIDTH; 1 uS EACH
        PLSEP   EQU     001H    ;(003C) STEP PULSE PERIOD; 50 uS EACH
        SMODE   EQU     000H    ;(0000) STEP MODE; 0 = NORMAL
        HDNU    EQU     008H    ;(0004) MAX. NUMBER OF DEVICES
        NHEAD   EQU     003H    ;(0003) NUMBER OF HEADS / DRIVE -1
```

```
        NCYL    EQU     0131H   ;(0098) NUMBER OF TRACKS / DEVICE -1
        RWCC    EQU     080H    ;(0080) REDUCED WRITE CURRENT TRACK
        DRIVE   EQU     000H    ;(0000) DRIVE TYPE
        NSEC    EQU     04CH    ;(004C) NUMBER OF SECTORS / TRACK
        MBLK    EQU     02D6CH  ;(2D6C) NO. BLOCKS / LOGICAL DEVICE
;
;       PORT ASSIGNMENTS ********************************************
;
        RXHD    EQU     0D8H    ;READ SASI PORT DATA
        TXHD    EQU     0D9H    ;WRITE SASI PORT DATA
        HSTAT   EQU     0DAH    ;READ CONTROL DATA
        SELHD   EQU     0DBH    ;WRITE SELECT BYTE
;
;       ERROR CODES *************************************************
;
;       OMTI 20L DRIVE ERRORS
;
        NOERR   EQU     00H     ;NO ERRORS
        NINDX   EQU     01H     ;NO INDEX
        NSEEK   EQU     02H     ;NO SEEK COMPLETE
        WRFLT   EQU     03H     ;WRITE FAULT
        DRVNR   EQU     04H     ;DRIVE NOT READY
        NTRK0   EQU     06H     ;NO TRACK 00 FOUND
;
;       OMTI 20L DATA ERRORS
;
        IDECC   EQU     10H     ;ECC ERROR IN ID FIELD
        DATER   EQU     11H     ;UNCORRECTABLE DATA ERROR
        NIDAD   EQU     12H     ;NO ADDRESS MARK FOUND IN ID
        NDAAD   EQU     13H     ;NO ADDRESS MARK FOUND IN DATA
        NOREC   EQU     14H     ;NO RECORD FOUND
        SEKER   EQU     15H     ;SEEK ERROR
        CORER   EQU     18H     ;CORRECTABLE DATA ERROR
        BDTRK   EQU     19H     ;BAD TRACK FLAG SET
        ALTER   EQU     1CH     ;ALTERNATE TRACK READ ERROR
;
;       OMTI 20L COMMAND ERRORS
;
        IVCMD   EQU     20H     ;INVALID COMMAND
        ILSEC   EQU     21H     ;ILLEGAL SECTOR ADDRESS
        VOLOF   EQU     23H     ;VOLUME OVERFLOW
;
;       OMTI 20L HARDWARE ERRORS
;
        RAMER   EQU     30H     ;RAM ERROR
        DMAER   EQU     33H     ;DMA TIMEOUT
;
;       HD HANDLER ERRORS
;
        IVUNT   EQU     07H     ;INVALID UNIT
        IVFCN   EQU     08H     ;INVALID FUNCTION CODE
        NOPWR   EQU     09H     ;NO POWER TO OMTI 20L
;
; ***** THE HANDLER ***********************************************
;
;       WHEN CALLED HL => BLOCK OF 8 BYTES, AS FOLLOWS:
;
;               FCN: DS 1  ;FUNCTION CODE
;               DRV: DS 1  ;DEVICE NUMBER
;               BLK: DS 2  ;FIRST BLOCK NO. FOR TRANSFER
;               BUF: DS 2  ;MEMORY POINTER FOR TRANSFER
;               XBC: DS 2  ;BYTE COUNT FOR TRANSFER
;
;       FUNCTION CODES:
;
;               - 4 = TURN OFF          NO ACTION
;               - 3 = SUBTRACT USER     NO ACTION
;           - 2 = ADD USER          NO ACTION
;           - 1 = GET SIZE          SET TBLK = MAX. NO. BLOCKS
;             Ø = READ DATA         READ DATA W/ ERROR CORRECTION
;             1 = WRITE DATA        WRITE WITH AUTOMATIC VERIFY
;             2 = VERIFY DATA       NO ACTION
;
        ORG     Ø5ØØØH          ;OR WHERE EVER YOU WISH
;
HDHAND: MVI     A,HDNU          ;MAX. NO. OF DEVICES
        CALL    HDPAR           ;GET PARAMETERS AND SETUP RETURN
        LXI     H,MBLK          ;BLOCKS PER LOGICAL DEVICE
        JP      REWRIT          ;IT IS A READ OR WRITE FUNCTION
        SHLD    TBLK            ;SAVE MAXIMUM BLOCKS
        POP     D               ;STRIP ERROR RETURN
        XRA     A               ;SET NO ERRORS
        RET                     ;ALL EXCEPT READ OR WRITE FUNCTIONS
;
;       GET TRANSFER DATA BYTES AND SETUP HANDLER RETURN ************
;
HDPAR:  POP     B               ;GET HANDLERS RETURN ADDR.
        LXI     D,HDRET         ;SET RETURN THROUGH ERROR ROUTINE
        PUSH    D
        PUSH    B               ;RESTORE RETURN ADDR.
        MOV     C,A             ;C = MAX. NO. OF DEVICES
        LXI     D,TFCN          ;POINT TO PARAMETER STORAGE
        MVI     B,ØØ8H          ;NO. BYTES IN PARAMETER BLOCK
        CALL    MOVDH           ;COPY INTO STORAGE AREA
        POP     H               ;RESTORE RETURN ADDRESS
        LDA     TDRV            ;DRIVE NO. ASKED FOR
        MVI     E,IVUNT         ;SET INVALID UNIT
        CMP     C               ;COMPARE TO MAX
        RNC                     ;IVU, RETURN TO ERROR
        LDA     TFCN            ;GET FUNCTION ASKED FOR
        CPI     2
        JC      OKDRV           ;READ OR WRITE
        MVI     E,IVFCN         ;SET INVALID FUNCTION
        CPI     -4              ;LOWEST POSSIBLE
        RC                      ;IVF, RETURN TO ERROR
OKDRV:  MVI     E,Ø             ;SET NO ERRORS
        ANA     A               ;TEST FUNCTION CODE
        PCHL                    ;TRICKY RETURN TO HDHAND
;
;       RETURN FROM PROCESSING ROUTINE  ******************************
;
HDRET:  JNZ     HDERR           ;ERRORS !!!!
        LXI     H,Ø             ;SUCCESSFUL TRANSFER
        SHLD    TBC             ;NO BYTES LEFT
        LHLD    BKCNT           ;GET NEXT BLOCK AFTER TRANSFER
        SHLD    TBLK            ;AND POINT AHEAD FOR NEXT PASS
        XRA     A               ;SET NO ERRORS
        RET
;
;       ERROR REPORTING ROUTINE             **************************
;
HDERR:  XRA     A
        STA     HDFLG           ;SET NO ADDITIONAL ERRORS
        STA     ECODE           ;SET NON VALID SECTOR ADDRESS
        MOV     A,E             ;CHECK FOR IVU
        CPI     IVUNT
        JZ      HDREX
        CPI     IVFCN           ;CHECK FOR IVF
        JZ      HDREX
;
;       GET ERROR CODE FROM CONTROLLER
;
ELOOP:  MVI     A,ØØ3H          ;REQUEST SENSE
        LXI     B,ØØØH          ;ERROR CORRECTION OK,RETRY OK
        CALL    COMND           ;DO IT
```

```
        CALL    DATOK           ;GET ERROR, SECTOR VALIDITY
        STA     ECODE
        CALL    DATOK           ;GET LOGICAL ADDR 2
        STA     SECTØ           ;SAVE ERROR SECTOR ADDRESS
        CALL    DATOK
        STA     SECT1
        CALL    DATOK
        STA     SECT2
        CALL    REASM           ;READ STATUS
        STA     HDFLG           ;INDICATE MORE ERRORS, IF SO
        LDA     ECODE
        ANI     Ø3FH            ;STRIP SECTOR VALIDITY
        MOV     E,A             ;ERROR CODE IN E
;
;       PRINT ERROR CODES TO SCREEN
;
HDREX:  LXI     H,ERTBL         ;FIND ERROR CODE IN TABLE
        LXI     B,4             ;TABLE ENTRY SIZE
HDERØ:  MOV     A,M
        ANA     A
        JZ      HDER1           ;USE UNKNOWN ?
        CMP     E
        JZ      HDER1           ;GOOD ERROR CODE
        DAD     B
        JMP     HDERØ           ;TRY NEXT CODE
HDER1:  INX     H               ;POINT TO ERROR MSG
        PUSH    H               ;SAVE POINTER
        LXI     H,ERMSG         ;SEND LEAD MESSAGE
        CALL    OSTR
        POP     H               ;POINTER BACK
        MVI     B,3             ;SEND ERROR MESSAGE
        CALL    PSTR
        LXI     H,HDNAM         ;SEND NAME
        MVI     B,2
        CALL    PSSTR
        LXI     H,TFCN          ;FUNCTION CODE
        MOV     B,M
        INX     H
        MOV     A,M             ;UNIT NO.
        CALL    LBYT            ;LIST UNIT
        CALL    PCOLN           ;COLON
        MOV     A,B
        CALL    LBYT            ;LIST FUNCTION
        LDA     ECODE
        ANI     Ø8ØH            ;VALID ADDRESS ?
        JZ      NOSEC           ;NO
        CALL    PSPAC           ;SPACE
        LDA     SECTØ
        CALL    LBYT            ;PRINT SECTOR ADDRESS
        LDA     SECT1
        CALL    LBYT
        LDA     SECT2
        CALL    LBYT
NOSEC:  CALL    CRLF
        LDA     HDFLG           ;MORE ERRORS ?
        ANA     A
        JNZ     ELOOP           ;YES
        STC                     ;INDICATE ERROR
        RET                     ;RETURN TO CALLER
;
DATOK:  IN      HSTAT           ;REQ, BSY, I/O MEANS READ DATA
        CPI     Ø15H
        JNZ     DATOK
        IN      RXHD
        RET
;
;       ERROR CODE TABLE
;
```

```
ERTBL:  DB      NSEEK,'ISK'     ;NO SEEK COMPLETE
        DB      WRFLT,'WFT'     ;WRITE FAULT
        DB      DRVNR,'DNR'     ;DRIVE NOT READY
        DB      IVUNT,'IVU'     ;INVALID UNIT
        DB      IVFCN,'IVF'     ;INVALID FUNCTION
        DB      NOPWR,'PWR'     ;NO POWER TO OMTI 20L
        DB      IDECC,'IDF'     ;ECC ERROR IN ID FIELD
        DB      DATER,'DAT'     ;UNCORRECTABLE DATA ERROR
        DB      NIDAD,'IDA'     ;NO ADDRESS MARK IN ID
        DB      NDAAD,'DAA'     ;NO ADDRESS MARK IN DATA
        DB      NOREC,'NRF'     ;NO RECORD FOUND
        DB      SEKER,'SKF'     ;SEEK FAILURE
        DB      CORER,'COR'     ;CORRECTABLE DATA ERROR
        DB      BDTRK,'BTK'     ;BAD TRACK FLAG SET
        DB      ALTER,'ALT'     ;ALTERNATE TRACK READ ERROR
        DB      ILSEC,'ILS'     ;ILLEGAL SECTOR ADDRESS
        DB      VOLOF,'VOF'     ;VOLUME OVERFLOW
        DB      Ø,    'UNK'     ;ALLOW OTHERS ?

;       READ AND WRITE ROUTINES         ****************************
;
;       SETUP FOR READ/ WRITE
;
REWRIT: IN      HSTAT           ;SEE IF POWERED UP
        ANI     Ø1FH            ;CONTROLL INPUTS
        CPI     Ø1FH            ;ALL HIGH ??
        MVI     E,NOPWR
        JZ      PRERR           ;YES, NO POWER TO CONTROLLER
        XRA     A
        STA     SECTØ           ;CLEAR DISK SECTOR ADDRESS
        STA     SECT1
        STA     SECT2
        STA     HALF2           ;SET NO MORE THAN 256 SECTORS
        CALL    IPARMS          ;SET PARAMETERS IF CPU RESET
        LHLD    TBC             ;GET NO. OF BYTES TO TRANSFER
        MOV     A,H
        ORA     L
        RZ                      ;NO BYTES, IGNORE BUT NO ERROR
        CALL    BC2BK           ;CONVERT HL = NO. BLOCKS
        MOV     A,C
        STA     LBLK            ;SAVE LAST BLOCK BYTE COUNT
        SHLD    BKCNT           ;SAVE TOTAL BLOCKS
        MOV     A,L             ;FIGURE BLOCKS TO TRANSFER
        STA     HALF1           ;LOWER 32K BYTES
        LXI     D,ØØ1ØØH        ;DE = 256
        CALL    CMPHD
        JZ      PROCS           ;256 BLOCKS EXACTLY
        JC      PROCS           ;LESS THAN 256 BLOCKS, OK
        CALL    SUBHD           ;SUBTRACT 256 BLOCKS
        MOV     A,L
        STA     HALF2           ;REMAINDER FOR SECOND PASS
PROCS:  LHLD    BKCNT           ;GET TOTAL BLOCKS TO READ
        XCHG
        LHLD    TBLK            ;GET BLOCK TO START
        PUSH    H               ;SAVE START BLOCK
        DAD     D               ;ADD BLOCKS TO READ
        SHLD    BKCNT           ;NEXT BLOCK AFTER TRANSFER
        LXI     D,MBLK          ;MAXIMUM ON DRIVE
        CALL    CMPHD           ;MORE THAN MAXIMUM ?
        JC      PROCX           ;OK !!!
        MVI     E,Ø23H          ;VOLUME OVERFLOW
        POP     H               ;STRIP START BLOCK
PRERR:  POP     H               ;STRIP NORMAL ERROR RETURN
        XRA     A
        STA     HDFLG           ;INDICATE THIS ERROR ONLY
        STA     ECODE           ;INDICATE INVALID SECTOR ADDRESS
        JMP     HDREX           ;SPECIAL ERROR EXIT
```

```
;       ADJUST SECTOR FOR RIGHT LOGICAL DEVICE
;
PROCX:  POP     H               ;RECOVER START BLOCK
        LDA     TDRV            ;GET DRIVE NUMBER
        ANA     A               ;CHECK FOR UNIT ZERO
        JZ      PROC2           ;HL = CORRECT SECTOR ADDRESS
PROCØ:  DAD     D               ;ADD MBLK TO NEXT UNIT
        JNC     PROC1           ;NOT OVER 64K SECTORS
        PUSH    H
        LXI     H,SECTØ         ;MSB'S OF DISK SECTORS
        INR     M
        POP     H
PROC1:  DCR     A
        JNZ     PROCØ
PROC2:  MOV     A,H             ;SAVE SECTOR ADDRESS
        STA     SECT1
        MOV     A,L
        STA     SECT2
        LDA     HALF2           ;TWO PASSES REQUIRED ?
        ANA     A
        JNZ     PROC4           ;MORE THAN 32K, DO TWO PASSES
;
;       READ/ WRITE LAST PASS
;
PROC3:  LDA     LBLK            ;LAST BLOCK BYTE COUNT
        CPI     Ø8ØH
        JZ      PROC4           ;DO A FULL BLOCK
        LDA     HALF1           ;SAVE ONE FOR SPECIAL TRAILING BYTES
        DCR     A
        STA     HALF1
        JZ      DOLAST          ;ONLY ONE BLOCK TO DO
PROC4:  LHLD    TMEM            ;GET MEMORY POINTER
        LDA     HALF1           ;BLOCKS TO READ/ WRITE
        MOV     C,A
        LDA     TFCN            ;GET FUNCTION CODE
        ANA     A
        JZ      RCODE           ;DO A READ
;
;       DO A WRITE WITH AUTOMATIC VERIFY
;
WCODE:  MVI     B,Ø1ØH          ;ENABLE VERIFY
        MVI     A,ØØAH          ;WRITE CODE
        CALL    COMND           ;SEND COMMAND
        CALL    WRITX           ;WRITE THE BLOCKS OUT
        RNZ                     ;ERROR, RETURN <NZ>
        JMP     TNEXT           ;SEE IF SECOND PART
;
;       DO A READ WITH ECC AND AUTOMATIC RETRY
;
RCODE:  MVI     B,ØØØH          ;ECC AND RETRY
        MVI     A,ØØ8H          ;READ CODE
        CALL    COMND           ;SEND COMMAND
        CALL    READX           ;READ THE BLOCKS
        RNZ                     ;ERROR
;
;       SETUP FOR SECOND PASS IF NECESSARY
;
TNEXT:  SHLD    TMEM            ;HL => NEXT MEM LOCATION
        LXI     H,HALF2         ;SEE IF ANY MORE BLOCKS
        MOV     A,M
        ANA     A
        JZ      DOLAST          ;WAS LESS THAN 32K
        STA     HALF1           ;SAVE NO. OF REAMINING
        MVI     M,Ø             ;NO MORE AFTER THIS
        LXI     H,SECT1         ;INCREMENT SECTOR POINTER
        INR     M               ;BY 256 SECTORS
        RNZ                     ;NO CARRY OUT
        DCX     H               ;POINT TO SECTØ
        INR     M               ;AND INCREMENT
        JMP     PROC3           ;RETURN TO READ/WRITE
;
;       READ/ WRITE TRAILING BYTES IN LAST BLOCK
;
DOLAST: LDA     LBLK            ;WAS LAST BLOCK FULL ?
        CPI     Ø8ØH
        MVI     E,Ø
        RZ                      ;DONE, NO ERRORS
        MOV     E,A             ;SAVE NO BYTES TO READ/WRITE
        LDA     HALF1           ;GET NO BLOCKS MOVED
        LXI     H,SECT2
        ADD     M               ;ADD TO SECTOR COUNT
        MOV     M,A
        JNC     DOL1
        DCX     H               ;INCREASE NEXT BITS
        INR     M
        JNZ     DOL1            ;DID NOT ROLL PAST ZERO
        DCX     H
        INR     M
DOL1:   LHLD    TMEM            ;GET BACK MEMORY POINTER
        MVI     C,1             ;ONLY ONE SECTOR TO READ/WRITE
        LDA     TFCN            ;READ OR WRITE ?
        ANA     A
        JZ      READL           ;READ
;       WRITE LAST BLOCK IN A FILE WITH TRAILING ZEROS
;
WRITL:  MVI     B,Ø1ØH          ;ENABLE VERIFY
        MVI     A,ØØAH          ;WRITE CODE
        CALL    COMND           ;SEND COMMAND
WRLØ:   IN      HSTAT           ;REQ, BSY MEANS WRITE DATA
        CPI     Ø14H
        JNZ     WRLØ
        MOV     A,M             ;WRITE DATA
        INX     H
        OUT     TXHD
        DCR     E               ;COUNTER
        JNZ     WRLØ
WRL1:   IN      HSTAT           ;REQ, BSY, C/D, I/O MEANS DONE
        CPI     Ø17H
        JZ      REASØ           ;READ STATUS AND RETURN
WRL2:   IN      HSTAT           ;REQ, BSY MEANS WRITE DATA
        CPI     Ø14H
        JNZ     WRL1
        XRA     A
        OUT     TXHD            ;WRITE A ZERO
        JMP     WRL2
;
;       READ LAST BLOCK OF FILE IGNORING TRAILING BYTES
;
READL:  MVI     B,ØØØH          ;ECC AND RETRY
        MVI     A,ØØ8H          ;READ CODE
        CALL    COMND           ;SEND COMMAND
RELØ:   IN      HSTAT           ;REQ, BSY, I/O MEANS READ
        CPI     Ø15H
        JNZ     RELØ
        IN      RXHD            ;GET BYTE
        MOV     M,A             ;PUT IN MEMORY
        INX     H
        DCR     E               ;COUNTER
        JNZ     RELØ
REL1:   IN      HSTAT           ;REQ, BSY, C/D, I/O MEANS DONE
        CPI     Ø17H
        JZ      REASØ           ;READ STATUS AND RETURN
REL2:   IN      HSTAT           ;REQ, BSY, I/O MEANS READ DATA
        CPI     Ø15H
        JNZ     REL1
        IN      RXHD            ;READ AND IGNORE
        JMP     REL2
;
;       INITIALIZE CONTROLLER   ************************************
;
IPARMS: LXI     H,RSFLG         ;HAS CPU RESET OCCURED ?
        XRA     A
        CMP     M
        RNZ                     ;NO
        INR     M               ;RESET DONE
        LXI     B,ØØØH
        MVI     A,ØC2H          ;ASSIGN PARAMETERS CODE
        CALL    COMND
        LXI     H,DTBL          ;PARAMETER LIST
        JMP     WRITX           ;WRITE THE DATA
;
;       DISK PARAMETER TABLE
;
DTBL:   DB      PLSEW           ;PULSE WIDTH
        DB      PLSEP           ;PULSE PERIOD
        DB      SMODE           ;STEP MODE
        DB      NHEAD           ;NUMBER OF R/W HEADS
        DB      NCYL SHR 8      ;MAX. TRACK ADDRESS HIGH
        DB      NCYL AND ØFFH   ;MAX. TRACK ADDRESS LOW
        DB      RWCC            ;REDUCED WRITE CURRENT TRACK
        DB      DRIVE           ;DRIVE TYPE
        DB      NSEC            ;SECTORS/ TRACK
        DB      ØØØH            ;RESERVED
;
;       SEND COMMAND TO CONTROLLER        ****************************
;
;       A     =  COMMAND CODE
;       SECTØ =  MSB'S OF SECTOR ADDRESS
;       SECT1 =       SECTOR ADDRESS
;       SECT2 =  LSB'S OF SECTOR ADDRESS
;       B     =  THE CONTROL FIELD
;       C     =  NUMBER OF SECTORS
;
COMND:  DI                      ;DISABLE INTERRUPTS
        PUSH    PSW             ;SAVE COMMAND
COMNØ:  IN      HSTAT           ;CHECK TO SEE IF BUSY OFF
        ANI     ØØ4H
        JNZ     COMNØ
        DCR     A               ;SELECT CONTROLLER ØFFH
        OUT     SELHD
        POP     PSW
        CALL    REQOK           ;SEND THE COMMAND
        LDA     SECTØ           ;MSB'S OF ADDRESS
        CALL    REQOK
        LDA     SECT1           ;HIGH ADDRESS
        CALL    REQOK
        LDA     SECT2           ;LSB'S OF ADDRESS
        CALL    REQOK
        MOV     A,C             ;NO. SECTORS
        CALL    REQOK
        MOV     A,B             ;CONTROL FIELD
REQOK:  PUSH    PSW             ;SAVE THE BYTE
REQØØ:  IN      HSTAT           ;REQ, BSY, C/D MEANS WRITE COMMAND
        CPI     Ø16H
        JNZ     REQØØ
        POP     PSW
        OUT     TXHD
        RET
```

```
;       READ DATA                                    ******************************
;
;       HL => MEMORY LOCATION FOR DATA
;
READX:  IN      HSTAT           ;REQ, BSY, C/D, I/O MEANS DONE
READØ:  CPI     Ø17H
        JZ      REASØ           ;GO GET STATUS AND MESSAGE
READ1:  IN      HSTAT           ;REQ, BSY, I/O MEANS READ DATA
        CPI     Ø15H
        JNZ     READØ
        IN      RXHD
        MOV     M,A
        INX     H
        JMP     READ1
;
;       WRITE DATA                                   ******************************
;
;       HL => MEMORY LOCATION WITH DATA
;
WRITX:  IN      HSTAT           ;REQ, BSY, C/D, I/O MEANS DONE
WRITØ:  CPI     Ø17H
        JZ      REASØ           ;GO GET STATUS AND MESSAGE
WRIT1:  IN      HSTAT           ;REQ, BSY MEANS WRITE DATA
        CPI     Ø14H
        JNZ     WRITØ
        MOV     A,M             ;WRITE DATA
        INX     H
        OUT     TXHD
        JMP     WRIT1
;
;       READ STATUS AND MESSAGE                      ******************************
;
REASM:  IN      HSTAT           ;REQ, BSY, C/D, I/O MEANS DONE
        CPI     Ø17H
        JNZ     REASM
REASØ:  IN      RXHD            ;GET THE STATUS BYTE
        MOV     E,A             ;SAVE IN E
REAS1:  IN      HSTAT           ;REQ, MSG, BSY, C/D I/O MEANS READ
        CPI     Ø1FH            ; THE MESSAGE BYTE
        JNZ     REAS1
        IN      RXHD            ;GET THE MESSAGE BYTE AND IGNORE
        MOV     A,E             ;TEST STATUS
        ANA     A
        EI                      ;ENABLE INTERRUPTS
        RET
```

## 'Tom Teaser' (Solution)

Tom Napier

Here was the puzzle; without using an MVI instruction or making any preconditions, load 06H into the A register using only two bytes.

This odd puzzle arose from my observation that though I have always used SUB A to clear the A register, nearly all the published programs I have seen use XRA A. At first glance the effect of the two instructions is exactly the same, but there is a subtle difference. Both instructions clear the Carry and Sign flags, and set the Parity and Zero flags. The difference lies in the Auxiliary Carry flag. XRA A clears it, but SUB A sets it. This is not a dramatic difference unless a DAA instruction follows. The sequence XRA A : DAA sets A = 0 and sets the zero flag. The sequence SUB A : DAA sets A = 6 and clears the zero flag. The prior content of the A register is immaterial.

This little exercise shows that the 8080 instruction set can still have some surprises. This particular one has been hidden from me in ten years of programming experience.

**LAST CHANCE!**

A limited number of copies of these two great books is available from:

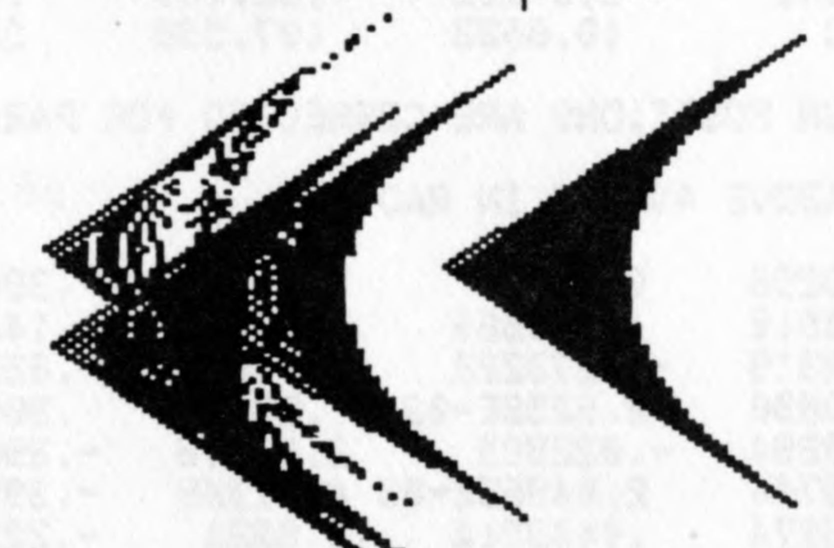
Doug Van Putte
18 Cross Bow Drive
Rochester, NY 14624

Price: $ 6.00 (USA)
$ 8.00 (overseas)

You haven't mastered your CCII if you have not worked through both these fine tutorials!

***HURRY!***

Color Graphics
for
INTECOLOR 3651 and
COMPUCOLOR II computers



by David B. Suits

# ASTRO

Wallace Rust
523 Britton Road
Greece, NY 14616

The ASTRO program, presented here, calculates the celestial positions of the sun, moon, and all eight planets. The sun and planets are located to within a few minutes of arc, and the moon within a degree or so. Writing such a computer program presents a number of interesting challenges, and ASTRO meets these with surprising results.

Most of the necessary constants and equations have been obtained from the excellent book by Peter Duffett-Smith, "Practical Astronomy with Your Calculator." (See full reference in program listing, lines 130-150.) This book presumes a scientific calculator or a computer with double precision arithmetic, and so a principle challenge has been to maintain accuracy while using Compucolor's single-precision BASIC.

Julian date is generally used by astronomers to refer to the time of events because it neatly avoids the peculiarities of the civil calendar. The Julian date is the number of days that have elapsed since January 1, 4713 B.C. If we want a precision of a few seconds per day, we will have Julian dates such as 2445538,1234 to work with. Therefore, in ASTRO we use an epoch (reference) of A.D. Jan 0,0 1980 instead, and count days forward or backward from that. Another way of preserving precision is to work in radians as much as possible, and convert radians to time or degrees only for the output.

Let us take a walk through the features of the program. At line 310 we dimension arrays for holding the orbital elements of the planets (PP), sun (SP), moon (MP), names (R$), and the calculated results (R). Lines 340-397 explain the subscripts.

Constants are defined in lines 400-450. They are entered to eight-digit precision because the Compucolor works to that precision internally. In lines 500-950 we load the arrays.

At lines 2000-2310 we input the observing site. You can add your own location to the program here, or you can type it in when the program runs by choosing "1" at the line 2200 prompt, which causes a branch to the input routine at lines 2400-2500. The date and time of observation are entered at lines 2505-2630.

ASTRO begins its calculations at line 2640, and they take about thirty seconds. Great care must be taken in calculating arctangents. The routine at lines 4100-4150 was written to do this correctly for all quadrants. You may have to modify the routine at lines 5000-5030 to suit your own printer; the routine in the listing is for the Radio Shack DMP-110.

Sample Printout of "ASTRO"

```
SKY POSITIONS FOR 1984 WED JUN  6

LOCAL STANDARD TIME:  22 HOURS, 0 MINUTES

FOR LAT.  40.72    LONG. WEST 74.02    ELEV.  10 METRES   (New York City)

DAYS SINCE JAN 0 THIS YEAR:  158

TIMES IN HOURS: LMT = 22              GMT = 3
                LST = 15.0463         GST = 19.9809


          ECLIPTIC      ECLIPTIC      RIGHT          DECLI-
          LONG.         LAT.          ASCENSION*     NATION*
BODY      (DEG)         (DEG)         (HOURS)        (DEG)

SUN         76.3728      0              5.01324       22.7443
MOON       169.316       5.00975       11.4359         8.25356
MERCURY     59.1972     -1.56545        3.82274       18.4519
VENUS       73.8518     -.150447        4.8336        22.3159
MARS       223.043      -1.31751       14.6783       -17.0109
JUPITER    280.602       .117435       18.7681       -22.9014
SATURN     221.261       2.53943       14.6425       -12.7943
URANUS     251.955       .0240665      16.6969       -22.2016
NEPTUNE    269.816       1.23014       17.9867       -22.2116
PLUTO      211.28        17.1477       14.3354         4.19362


          HOUR ANGLE* AZIMUTH*      ALTITUDE*      PHASE
BODY      (DEG WEST)  (DEG)         (DEG)          (0 TO 1)

SUN         150.496     330.918      -20.8615       1
MOON        54.1552     251.446       32.2           .525819
MERCURY     168.353     347.245      -29.8454        .764606
VENUS       153.19      333.211      -22.2139        .999087
MARS        5.51894     186.229       32.0417        .964916
JUPITER     304.173     129.689       7.94875        .998402
SATURN      6.05631     187.325       36.1923        .99913
URANUS      335.24      155.087       22.9982        .999996
NEPTUNE     315.893     138.181       14.9048        .999985
PLUTO       10.6632     197.538       52.2351        .999842

* MOON POSITIONS ARE CORRECTED FOR PARALLAX.

THE ABOVE ANGLES IN RADIANS:

 1.33296    0            1.31246    .396963     2.62664     5.77561    -.364102
 2.95512    .0874366     2.99392    .144052     .945186     4.38856     .561996
 1.03319   -.0273222     1.00079    .322047     2.93831     6.06057    -.5209
 1.28896   -2.6258E-03   1.26543    .389486     2.67367     5.81563    -.387705
 3.89284   -.022995      3.84278   -.296897     .0963237    3.2503      .559233
 4.89744    2.04963E-03  4.91348   -.399704     5.30881     2.26349     .138732
 3.86174    .0443214     3.8334    -.223302     .105703     3.26943     .631675
 4.39745    4.2004E-04   4.37124   -.387491     5.85105     2.70678     .401395
 4.70917    .02147       4.70892   -.387666     5.51337     2.4117      .260138
 3.68753    .299283      3.753      .0731926    .186109     3.44768     .911674
```

Calculation routines are headed by remarks and by references in brackets to the Duffett-Smith book. In the routines at lines 16500-16960 we take into account the parallax caused by the nearness of the moon. (The only factor not taken into account in this program is the precession of the earth.) At lines 17140-17300 we calculate look-angles for the chosen observing site.

At line 20000 we output the results to the screen as three pages of tables. At the user's option (line 20760), results can also be directed to the printer. The routine at line 60000 permits a printed listing by typing "GOTO 60000" in 'immediate' mode.

The sample output for a particular time and site will allow you to debug your program. If any reader wants a listing of the 107 variables and their meaning, send me a 3 by 9 inch SASE. Ten dollars (US) will purchase both the list and a diskette in CCII v6.78 format. This program uses about 12K of RAM.

```
100 REM  "ASTRO", CALC & PRINT ASTRONOMICAL POSITIONS.
105 REM  WALLACE R. RUST, 523 BRITTON ROAD, GREECE, NY 14616.
110 REM  VERSION: 9 JAN 1985
125 REM
130 REM  REF: "PRACTICAL ASTRONOMY WITH YOUR CALCULATOR",
140 REM  2ND EDITION, BY PETER DUFFETT-SMITH,
145 REM  CAMBRIDGE UNIVERSITY PRESS, (C) 1981.
150 REM  SECTION REFERENCES IN [ ].
160 REM
170 REM  LAMBDA, BETA = ECLIPTIC LONGITUDE, LATITUDE
171 REM  ALPHA, DELTA = RIGHT ASCENSION, DECLINATION
172 REM  EPOCH JAN 0.0 1980
173 REM
180 REM  CHANGE LINES 5000-5110 TO SUIT YOUR OWN PRINTER!
185 REM  ENTER YOUR FAVORITE STATIONS AT LINES 2000-2390!
190 REM
200 REM  X--- TITLE PAGE
202 CLEAR 200
205 PLOT 6,3,12,14
210 PRINT TAB( 26);:PLOT 6,38:PRINT " > ASTRO < "
220 PLOT 6,3,15:PRINT
230 PRINT "THIS PROGRAM CALCULATES POSITIONS OF THE SUN, MOON, AND PLANETS."
240 PLOT 6,5:PRINT "TURN ON THE PRINTER!"
300 REM  X--- HOUSEKEEPING
310 DIM PP(8,6),SP(5),MP(7),R$(9),R(9,7)
330 REM
340 REM  PP(I,J):
350 REM    I=0 EARTH      J=0 PERIOD
352 REM        1 MERCURY      1 LONG. AT EPOCH
354 REM        2 VENUS        2 LONG. AT PERIH.
356 REM        3 MARS         3 ECCENTRICITY
358 REM        4 JUPITER      4 SEMI-MAJ AXIS (AU)
360 REM        5 SATURN       5 INCLINATION
362 REM        6 URANUS       6 LONG. ASCEN. NODE
364 REM        7 NEPTUNE
366 REM        8 PLUTO
368 REM
380 REM  R(I,J):
382 REM    I=0 SUN        J=0 LAMBDA
384 REM        1 MOON         1 BETA
386 REM        2 MERCURY      2 ALPHA
388 REM        3 VENUS        3 DELTA
390 REM        4 MARS         4 HOUR ANGLE
392 REM        5 JUPITER      5 AZIMUTH
394 REM        6 SATURN       6 ALTITUDE
395 REM        7 URANUS       7 PHASE
396 REM        8 NEPTUNE
397 REM        9 PLUTO
398 REM
400 PI= 3.1415926:P2= 6.2831853:DR= 57.29578:HR= 3.8197186
410 KA= 360:KB= 365.2422:KC= 13.1763966:KD= .1114041
420 KE= .0529539:KF= 1.2739:KG= .1858:KH= .37
430 KI= 6.2886:KJ= .214:KK= .6583:KL= .16
440 KM= .001:KN= .065709822:KQ= 1.002738
450 KR= .996647:KS= 6378140:KT= 6378.16:REM  [#35,36]
500 REM  X--- SUN ELEMENTS [P. 82;44]
510 RESTORE 520
520 DATA 278.83354,282.596403,.016718,1.495985E8,.533128
521 DATA 23.441884
```

```
520 DATA 278.83354,282.596403,.016718,1.495985E8,.533128
521 DATA 23.441884
530 FOR J= 0TO 5:READ SP(J):NEXT
620 DATA 64.975464,349.383063,151.950429,5.145396,.0549,.5181,384401,.9507
630 FOR J= 0TO 7:READ MP(J):NEXT
700 REM  ⧖--- EARTH, MERC, VEN, MARS-PLUTO ELEMENTS [P. 100]
705 RESTORE 710
710 DATA 1.00004,98.83354,102.5964,.016718,1,0,0
720 DATA .24085,231.2973,77.144213,.2056306,.3870986,7.0043579,48.094173
730 DATA .61521,355.73352,131.28958,.0067826,.7233316,3.394435,76.499752
750 DATA 1.88089,126.30783,335.690816,.0933865,1.5236883,1.8498011,49.4032
760 DATA 11.86224,146.966365,14.009549,.0484658,5.202561,1.3041819,100.25202
770 DATA 29.45771,165.32224,92.665397,.0556155,9.554747,2.4893741,113.48883
775 DATA 84.01247,228.07085,172.73633,.0463232,19.21814,.7729895,73.876864
780 DATA 164.79558,260.3579,47.867215,.0090021,30.10957,1.7716017,131.56065
785 DATA 250.9,209.439,222.972,.25387,39.78459,17.137,109.941
800 FOR I= 0TO 8:FOR J= 0TO 6:READ PP(I,J):NEXT :NEXT
900 REM  ⧖--- NAMES
910 RESTORE 920
920 DATA "SUN","MOON","MERCURY","VENUS","MARS","JUPITER"
921 DATA "SATURN","URANUS","NEPTUNE","PLUTO"
930 FOR J= 0TO 9:READ R$(J):NEXT
940 W$= "SUNMONTUEWEDTHUFRISAT"
950 M$= "JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC"
2000 REM  ⧖--- INPUT SITE
2010 PLOT 6,2:PRINT :PRINT "AVAILABLE SITES: ":PLOT 6,6
2020 PRINT " 1. -USER'S CHOICE-"
2030 PRINT " 2. GREENWICH (LONDON)"
2040 PRINT " 3. NEW YORK, NY"
2050 PRINT " 4. MIAMI, FL"
2060 PRINT " 5. ERIE, PA"
2070 PRINT " 6. GREECE, NY (RUST'S HOME)"
2080 PRINT " 7. SAN DIEGO, CA"
2090 REM  ADD  #8 HERE
2100 REM  ADD  #9 HERE
2110 REM  ADD #10 HERE
2120 REM
2200 PRINT :PLOT 6,1:INPUT "ENTER YOUR CHOICE (1 TO 6): ";Q
2210 IF Q< 1OR Q> 10THEN 2200
2220 ON QGOTO 2400,2302,2303,2304,2305,2306,2307,2308,2309,2310
2230 REM  SY=LATITUDE; SX=LONGITUDE; SE=ELEV IN METRES; TZ=TIME ZONES WESTWARD
2302 SY= 51.5:SX= 0:SE= 10:TZ= 0:GOTO 2500
2303 SY= 40.72:SX= 74.02:SE= 10:TZ= 5:GOTO 2500
2304 SY= 25.77:SX= 80.2:SE= 5:TZ= 5:GOTO 2500
2305 SY= 42.13:SX= 80.06:SE= 200:TZ= 5:GOTO 2500
2306 SY= 43.23625:SX= 77.63894:SE= 113:TZ= 5:GOTO 2500
2307 SY= 32.76:SX= 117.22:SE= 10:TZ= 8:GOTO 2500
2308 REM  ADD  #8 HERE
2309 REM  ADD  #9 HERE
2310 REM  ADD #10 HERE
2390 REM
2400 PRINT :PLOT 6,3
2410 INPUT "ENTER LATITUDE IN DEGREES:  ";SY
2420 IF SY< - 90OR SY> 90THEN 2410
2430 INPUT "ENTER LONGITUDE IN DEGREES WEST OF GREENWICH: ";SX
2440 IF SX< 0OR SX> 360THEN 2430
2450 INPUT "ENTER ELEVATION IN METRES ABOVE SEA LEVEL: ";SE
2470 INPUT "ENTER TIME ZONE NUMBER (0 TO 23): ";TZ
2480 TZ= INT (TZ):IF TZ< 0OR TZ> 23THEN 2470
2500 LA= SY/ DR
2505 REM  ⧖--- INPUT DATE & TIME
2510 PLOT 6,5:PRINT :PRINT "ENTER DATE OF INTEREST:":PLOT 6,3
2520 INPUT "YEAR (1583 TO 2400):   ";TY
2530 TY= INT (TY):IF TY< 1583OR TY> 2400THEN 2520
2540 INPUT "MONTH (1 TO 12):       ";TM
2550 TM= INT (TM):IF TM< 1OR TM> 12THEN 2540
2560 INPUT "DATE (1 TO 31):        ";TD
2570 TD= INT (TD):IF TD< 1OR TD> 31THEN 2560
2580 PLOT 6,5:PRINT "ENTER LOCAL STANDARD TIME OF INTEREST:":PLOT 6,3
2590 INPUT "HOUR (0 TO 23):        ";TH
2600 TH= INT (TH):IF TH< 0OR TH> 23THEN 2590
2610 INPUT "MINUTE (0 TO 60.0):    ";TN
2620 IF TN< 0OR TN> 60THEN 2610
2630 LM= TH+ TN/ 60
2640 PRINT "⧖WAIT..."
```

```
2700 REM  ⧖--- CALC DAYS SINCE JAN 0.0 1980
2701 REM  [COMPUCOLOR 'BIORHYTHMS']
2710 D1= TH/ 24+ TN/ 1440
2720 M9= (- 1)* INT (((14- TM)/ 12)+ KM)
2730 J1= TD- 2447095+ INT ((1461* (TY+ 4800+ M9)/ 4)+ KM)
2740 J2= J1+ INT ((367* (TM- 2- 12* M9)/ 12)+ KM)
2750 J1= J2- INT ((3* (TY+ 4900+ M9)/ 400)+ KM)
2760 WD= J1- 7* INT ((J1/ 7)+ KM)+ 1:WD= INT (WD+ KM)
2770 DE= J1- 29219+ D1
2800 REM  ⧖--- CALC WHOLE DAYS SINCE JAN 0.0 THIS YEAR
2801 REM  [W. RUST]
2810 D$= "000031059090120151181212243273304334"
2820 DJ= VAL (MID$ (D$,3* TM- 2,3))+ TD
2830 LY= 0:IF TY/ 4= INT (TY/ 4)THEN LY= 1
2840 IF TY/ 400= INT (TY/ 400)THEN LY= 0
2850 IF TM> 2THEN DJ= DJ+ LY
3000 REM  ⧖--- LMT TO GMT [#9]
3010 GM= LM+ TZ:IF GM> = 24THEN GM= GM- 24
3030 REM  ⧖--- CALC GST AT JAN 0.0 THIS YEAR [#4,12]
3035 Y= TY- 1:A= INT (Y/ 100):B= 2- A+ INT (A/ 4)
3040 C= INT (365.25* Y):S= B+ C- 693597.5:T= S/ 36525
3045 R= 6.6460656+ 2400.051262* T+ .00002581* T* T
3047 B= 24* TY- 45576- R
3050 REM  ⧖--- GMT TO GST [#12]
3070 T0= KN* DJ- B
3080 X= T0+ KQ* GM:GOSUB 4050:GS= X
3100 REM  ⧖--- GST TO LST [#14]
3110 LS= GS- SX/ 15:IF LS< 0THEN LS= LS+ 24
3999 GOTO 10000
4000 REM  ⧖--- SUBR X=MOD(X,360)
4010 IF X< 0THEN X= X+ KA:GOTO 4010
4020 IF X> = KATHEN X= X- KA:GOTO 4020
4030 RETURN
4050 REM  ⧖--- SUBR X=MOD(X,24)
4060 IF X< 0THEN X= X+ 24:GOTO 4060
4070 IF X> = 24THEN X= X- 24:GOTO 4070
4080 RETURN
4100 REM  ⧖--- SUBR A=ATN(Y/X) RADIANS
4110 IF X< > 0THEN 4140
4120 A= PI/ 2:IF Y< 0THEN A= 3* PI/ 2
4130 RETURN
4140 A= ATN (Y/ X):IF X< 0THEN A= A+ PI:RETURN
4150 IF Y< 0THEN A= A+ P2
4160 RETURN
4200 REM  ⧖--- SUBR X=MOD(X,2*PI)
4210 IF X< 0THEN X= X+ P2:GOTO 4210
4220 IF X> = P2THEN X= X- P2:GOTO 4220
4230 RETURN
4300 REM  ⧖--- SUBR OUTPUT PAGING
4310 IF Q$= "P"THEN RETURN
4320 INPUT "⧖PRESS ⧖<RETURN> ⧖FOR MORE..";A$:PLOT 6,3:RETURN
5000 REM  ⧖--- SUBR SELECT PRINTER
5010 POKE 33289,80:PLOT 15,27,18,4,27,13:OUT 8,4
5020 PLOT 30,19,27,26,27,19:REM  PRINTER FONT
5030 RETURN
5100 REM  ⧖--- SUBR SELECT CRT
5110 OUT 8,255:POKE 33265,0:POKE 33289,64:RETURN
9999 REM
 10000 REM  ⧖--- SUN CALC [#42]
 10010 X= KA* DE/ KB:GOSUB 4000:N= X
 10020 X= N+ SP(0)- SP(1):GOSUB 4000:SM= X/ DR
 10030 E= KA* SP(2)* SIN (SM)/ PI
 10040 X= N+ E+ SP(0):GOSUB 4000:SL= X/ DR
 10050 R(0,0)= SL:R(0,1)= 0:REM  SUN LAMBDA, BETA (RAD)
 10060 R(0,7)= 1
 11000 REM  ⧖--- MOON CALC [#61]
 11010 X= KC* DE+ MP(0):GOSUB 4000:L= X
 11020 X= L- KD* DE- MP(1):GOSUB 4000:MM= X/ DR
 11030 X= MP(2)- KE* DE:GOSUB 4000:N= X
 11040 C= L/ DR- SL:EV= KF* SIN (2* C- MM)
 11050 AE= KG* SIN (SM):A3= KH* SIN (SM)
 11060 X= MM+ (EV- AE- A3)/ DR:MM= X:REM  CORRECTED ANOMALY
 11070 EC= KI* SIN (MM)
 11080 A4= KJ* SIN (2* MM)
 11090 L= L+ EV+ EC- AE+ A4
```

```
11095 D= L/ DR- SL
11100 V= KK* SIN (2* D)
11110 L= L+ V
11120 N= N- KL* SIN (SM)
11125 J1= (L- N)/ DR
11130 Y= SIN (J1)* COS (MP(3)/ DR)
11140 X= COS (J1)
11150 GOSUB 4100:X= A+ N/ DR:GOSUB 4200:R(1,0)= X:REM  MOON LAMBDA (RAD)
11160 J2= SIN (J1)* SIN (MP(3)/ DR)
11170 R(1,1)= ATN (J2/ SQR (1- J2* J2)):REM  MOON BETA (RAD)
11180 REM  [#63]
11190 R(1,7)= (1- COS (D))/ 2:REM  MOON PHASE
12000 REM  I--- EARTH CALC [#50]
12010 X= KA* DE/ KB/ PP(0,0):GOSUB 4000:NE= X
12020 ME= NE+ PP(0,1)- PP(0,2)
12030 X= NE+ KA* PP(0,3)* SIN (ME/ DR)/ PI+ PP(0,1):GOSUB 4000:LE= X
12040 VE= LE- PP(0,2)
12050 RE= (1- PP(0,3)* PP(0,3))/ (1+ PP(0,3)* COS (VE/ DR))
13000 REM  I--- BEGIN MERCURY THRU PLUTO CALC [#50]
13005 FOR J= 1TO 8
13010 X= KA* DE/ KB/ PP(J,0):GOSUB 4000:NP= X
13020 MP= NP+ PP(J,1)- PP(J,2)
13030 X= NP+ KA* PP(J,3)* SIN (MP/ DR)/ PI+ PP(J,1):GOSUB 4000:LP= X
13040 VP= LP- PP(J,2)
13050 RP= PP(J,4)* (1- PP(J,3)* PP(J,3))/ (1+ PP(J,3)* COS (VP/ DR))
13060 REM
13065 J9= (LP- PP(J,6))/ DR
13070 J1= SIN (J9)* SIN (PP(J,5)/ DR)
13080 PS= ATN (J1/ SQR (1- J1* J1))
13090 Y= SIN (J9)* COS (PP(J,5)/ DR)
13100 X= COS (J9)
13110 GOSUB 4100:LP= A* DR+ PP(J,6)
13120 RP= RP* COS (PS)
13200 IF J> 2THEN 15000
14000 REM  I--- CONTINUE MERC & VENUS CALC [#50]
14010 J9= (LE- LP)/ DR
14020 Y= RP* SIN (J9):X= RE- RP* COS (J9)
14030 GOSUB 4100
14040 X= 180+ LE+ A* DR:GOSUB 4000:LD= X/ DR
14041 R(J+ 1,0)= LD:REM  PLANET LAMBDA (RAD)
14045 D= LD- LP/ DR
14050 Y= RP* TAN (PS)* SIN (D)
14060 X= RE* SIN (- J9)
14070 A= ATN (Y/ X):R(J+ 1,1)= A:REM  PLANET BETA (RAD)
14080 REM  [#54]
14090 R(J+ 1,7)= (1+ COS (D))/ 2:REM  PLANET PHASE
14100 NEXT J
15000 REM  I--- CONTINUE MARS-TO-PLUTO CALC [#50]
15010 J9= (LP- LE)/ DR
15020 Y= RE* SIN (J9):X= RP- RE* COS (J9)
15030 GOSUB 4100:X= A* DR+ LP:GOSUB 4000
15035 LD= X/ DR:R(J+ 1,0)= LD:REM  PLANET LAMBDA (RAD)
15040 D= LD- LP/ DR
15050 Y= RP* TAN (PS)* SIN (D):X= RE* SIN (J9)
15060 A= ATN (Y/ X):R(J+ 1,1)= A:REM  PLANET BETA (RAD)
15065 REM  [#54]
15070 R(J+ 1,7)= (1+ COS (D))/ 2:REM  PLANET PHASE
15080 NEXT J
15100 REM  WE NOW HAVE LAMBDA & BETA FOR ALL BODIES
16000 REM  I--- CONVERT (LAMBDA,BETA) TO (ALPHA,DELTA) [#27]
16090 EP= SP(5)/ DR
16100 FOR I= 0TO 9
16110 J1= SIN (R(I,1))* COS (EP)+ COS (R(I,1))* SIN (EP)* SIN (R(I,0))
16120 R(I,3)= ATN (J1/ SQR (1- J1* J1)):REM  DELTA (RAD)
16125 REM
16130 Y=.SIN (R(I,0))* COS (EP)- TAN (R(I,1))* SIN (EP)
16140 X= COS (R(I,0))
16150 GOSUB 4100:R(I,2)= A:REM  ALPHA (RAD)
16160 NEXT I
16300 REM  I--- CALC HOUR ANGLE [#24]
16320 FOR I= 0TO 9
16330 X= LS- R(I,2)* HR:GOSUB 4050:HA= X/ HR
16340 R(I,4)= HA:REM  HOUR ANGLE (RAD)
16350 NEXT I
16500 REM  I--- MOON PARALLAX [#65]
```

```
16510 Y= 1- MP(4)* MP(4)
16511 X= 1+ MP(4)* COS (MM+ EC/ DR):PA= Y/ X
16520 PK= PA* MP(6):PH= MP(7)/ PA/ DR
16800 REM  %--- MOON PARALLAX [#35]
16810 U= ATN (KR* TAN (LA)):H= SZ/ KS
16820 PB= KR* SIN (U)+ H* SIN (LA):PC= COS (U)+ H* COS (LA)
16900 REM  %--- MOON PARALLAX [#36]
16910 HA= R(1,4):R= 1/ SIN (PH)
16920 Y= PC* SIN (HA):X= R* COS (R(1,3))- PC* COS (HA)
16930 D= ATN (Y/ X):H1= HA+ D
16935 R(1,4)= H1:REM  CORR. MOON HA
16940 R(1,2)= R(1,2)- D:REM  CORR. MOON ALPHA
16950 Y= R* SIN (R(1,3))- PB:X= R* COS (R(1,3))* COS (HA)- PC
16960 R(1,3)= ATN (COS (H1)* Y/ X):REM  CORR. MOON DELTA
17140 REM  %--- CONVERT (ALPHA,DELTA) TO (AZ,ALT) [#25]
17150 FOR I= 0TO 9
17155 HA= R(I,4)
17160 J1= SIN (R(I,3))* SIN (LA)+ COS (R(I,3))* COS (LA)* COS (HA)
17170 AL= ATN (J1/ SQR (1- J1* J1)):R(I,6)= AL:REM  ALTITUDE (RAD)
17180 REM
17190 Y= SIN (R(I,3))- SIN (LA)* J1
17200 X= COS (LA)* COS (AL):J2= Y/ X
17210 A= ATN (SQR (1- J2* J2)/ J2)
17220 IF J2< 0THEN A= A+ PI
17221 IF HA< PITHEN A= P2- A
17230 R(I,5)= A:REM  AZIMUTH (RAD)
17300 NEXT I
20000 REM  %--- PRINT RESULTS
20010 PLOT 12
20020 Q$= "S":REM  RESULTS TO SCREEN FIRST
20030 IF Q$= "P"THEN GOSUB 5000
20080 W1$= MID$ (W$,WD* 3- 2,3)
20090 M1$= MID$ (M$,TM* 3- 2,3)
20100 PRINT "SKY POSITIONS FOR";TY;" ";W1$;" ";M1$;" ";TD
20110 PRINT
20120 PRINT "LOCAL STANDARD TIME: ";TH;" HOURS,";TN;" MINUTES"
20130 PRINT
20140 PRINT "FOR LAT. ";SY;"    LONG. WEST";SX;"    ELEV. ";SE;" METRES"
20150 PRINT
20160 PRINT "DAYS SINCE JAN 0 THIS YEAR: ";DJ
20170 PRINT
20180 PRINT "TIMES IN HOURS: LMT =";LM;TAB( 36);"GMT =";GM
20190 PRINT "                LST =";LS;TAB( 36);"GST =";GS
20200 PRINT :PRINT
20210 PRINT "           ECLIPTIC    ECLIPTIC    RIGHT       DECLI-"
20220 PRINT "           LONG.       LAT.        ASCENSION*  NATION*"
20230 PRINT "BODY       (DEG)       (DEG)       (HOURS)     (DEG)"
20240 PRINT
20300 FOR I= 0TO 9
20310 PRINT R$(I);TAB( 12);R(I,0)* DR;TAB( 24);R(I,1)* DR;
20320 PRINT TAB( 36);R(I,2)* HR;TAB( 48);R(I,3)* DR
20350 NEXT I
20500 PRINT :PRINT
20510 GOSUB 4300
20520 PRINT "           HOUR ANGLE* AZIMUTH*    ALTITUDE*   PHASE"
20530 PRINT "BODY       (DEG WEST)  (DEG)       (DEG)       (0 TO 1)"
20540 PRINT
20600 FOR I= 0TO 9
20610 PRINT R$(I);TAB( 12);R(I,4)* DR;TAB( 24);R(I,5)* DR;
20620 PRINT TAB( 36);R(I,6)* DR;TAB( 48);R(I,7)
20650 NEXT I
20660 PRINT :PRINT "* MOON POSITIONS ARE CORRECTED FOR PARALLAX.":PRINT
20670 GOSUB 4300
20700 PRINT "THE ABOVE ANGLES IN RADIANS:":PRINT
20710 FOR I= 0TO 9
20720 FOR J= 0TO 6
20730 PRINT TAB( J* 11.5);R(I,J);
20740 NEXT J:PRINT :NEXT I:PRINT :PRINT
20750 IF Q$= "P"THEN 30000
20760 INPUT "%DO YOU WANT ABOVE RESULTS SENT TO PRINTER? (%Y %OR %N%) ";A$
20770 IF A$= "Y"THEN Q$= "P":GOTO 20030
30000 GOSUB 5100:GOTO 2000
30010 END
60000 REM  %--- LIST ON PRINTER
60010 GOSUB 5000:LIST :PRINT :GOSUB 5100:END
```

# *Wator*

A spectacular assembly language presentation.

Tom Napier
12 Birch Street
Monsey, NY 10952

*This program is an assembly language implementation of a program suggested by A. K. Dewdney in the December, 1984 issue of "The Scientific American" (pp 14). The toroidal planet, WA-TOR, is populated by sharks and fish. By establishing their initial numbers, their breeding times, and the nourishment required by the sharks, an ecological universe, having its own special rhythm, is created. The computer program plots before you, on the CRT, the evolution of this universe in terms of the populations of its two species of inhabitants.*

*The assembly language code is commented sparingly and will not take long to type. Assembly has been ORGed at 8200H. The graphics character set is required. At run time, the following screen display will be presented, and you must set up the initial parameters for the ecology of WA-TOR. The numbers in parentheses are a suggested starting point.*

```
WELCOME TO THE WATERY WORLD OF WA-TOR.

YOU MAY SELECT ITS POPULATION.

ENTER NUMBER OF FISHES ? (500)
ENTER NUMBER OF SHARKS ? (20)
ENTER FISH BREEDING TIME (1-7) ? (3)
ENTER SHARK BREEDING TIME (1-7) ? (6)
ENTER TIME TO STARVE A SHARK (1-7) ? (4)
```

*Now the display begins. Upper left numbers show the number of fish, followed by the number of sharks. Fish are in cyan, sharks in red, all on a blue background. The screen redraws itself about once each second.*

*The suggested starting parameters appear at first to quickly diminish the fish population, but, within the first minute, you will observe that Nature has a way of striking a balance. In order to appreciate the operation and theory behind the concept demonstrated in this program, you should read the article by Mr. Dewdney, who presents a lucid and informative description of its performance. You will also share my admiration for the rendition that Tom Napier has presented. [ED]* □

```
;       SHARKS AND FISHES PROGRAM  22/12/84

;       COPYRIGHT (C) 1984   T. M. NAPIER

;       AN IMPLEMENTATION ON THE COMPUCOLOR II OF THE
;       PROGRAM "WA-TOR" DESCRIBED BY A. K. DEWDNEY
;       IN THE DECEMBER 1984 SCIENTIFIC AMERICAN.

;       DISPLAY SIZE 64 BY 32

;       2K STORAGE, ONE BYTE PER LOCATION
;       BYTE CONTAINS  M F SSS AAA

;               M = 1 IF OBJECT ALREADY MOVED
;               F = 0 IF FISH, 1 IF SHARK
;               SSS = TIME SINCE SHARK LAST ATE
;               AAA = TIME SINCE LAST BREEDING

;               IF FISH, SSS IS NOT NEEDED SO SSS := 100
;               THUS A ZERO BYTE EQUALS AN EMPTY SPACE

;               INITIAL FISH = 2XH, X = RANDOM AGE
;               INITIAL SHARK = 4XH

ORG     8200H

WTOR:   LXI     SP,STK
        MVI     A,195
        STA     UINP
        LXI     H,INP
        SHLD    UINP+1
        MVI     A,31
        STA     KBFL
        CALL    STR
        DB      6,2,15,12,10,10,0
        CALL    STR
        DB      'WELCOME TO THE WATERY WORLD OF WA-TOR.',13,10,10,0
        CALL    STR
        DB      'YOU MAY SELECT ITS POPULATION.',13,10,10,0
        CALL    STR
        DB      'ENTER NUMBER OF FISHES ',0
        CALL    NUMB
        SHLD    NOF
        CALL    STR
        DB      'ENTER NUMBER OF SHARKS ',0
        CALL    NUMB
        SHLD    NOS
        CALL    STR
        DB      'ENTER FISH BREEDING TIME (1-7) ',0
        CALL    NUMB
        STA     FBA
        CALL    STR
        DB      'ENTER SHARK BREEDING TIME (1-7) ',0
        CALL    NUMB
        STA     SBA
        CALL    STR
        DB      'ENTER TIME TO STARVE A SHARK (1-7) ',0
        CALL    NUMB
        ADD     A
        ADD     A
        ADD     A               ;MULTIPLY BY 8
        STA     SSA
        CALL    STR
        DB      6,34,12,0
        SUB     A
        STA     KBFL
        CALL    SETU
WTR1:   CALL    DISP
        CALL    SCOR
        CALL    FSCA
        CALL    SSCA
        LDA     RDY
        CPI     50H
        JNZ     WTR1
        LXI     SP,8042H
        LXI     H,38H
        PUSH    H
        LXI     H,KBFL
        JMP     ESCD
```

```
INP:    LXI     H,IBUF
        MOV     A,E
        CPI     13
        JZ      ICR
        CPI     26
        JZ      IBS
        MOV     A,M
        CPI     4
        RNC
        INR     A
        MOV     M,A
        ADD     L
        MOV     L,A
        MOV     M,E
        MOV     A,E
        CALL    TTO
        RET

ICR:    MOV     A,M
        MVI     M,5
        INR     A
        ADD     L
        MOV     L,A
        MOV     M,E
        CALL    CRLF
        RET

IBS:    MOV     A,M
        DCR     A
        RM
        MOV     M,A
        MVI     A,26
        CALL    TTO
        MVI     A,' '
        CALL    TTO
        MVI     A,26
        CALL    TTO
        RET

NUMB:   LXI     H,IBUF
        MVI     M,0
        XCHG
        LXI     H,0
NUM1:   LDAX    D
        CPI     5
        JNZ     NUM1            ;WAIT TILL NUMBER INPUT
NUM2:   INX     D
        LDAX    D
        CPI     13
        JZ      NUM3
        MOV     B,H
        MOV     C,L
        DAD     H
        DAD     H
        DAD     B
        DAD     H               ;HL = 10*HL
        SUI     30H             ;DIGIT TO NUMBER
        CALL    ADHL
        JMP     NUM2

NUM3:   MOV     A,L
        ANI     7
        RET

SETU:   LDA     TSEC
        MOV     L,A
        LDA     TMIN
        MOV     H,A
        SHLD    RANN            ;RANDOMIZE
        MVI     A,19
        STA     RCYC            ;RANDOMIZER CYCLE COUNT
```

```
        CALL    CLR             ;CLEAR SEA
        LHLD    NOF
        MOV     B,H
        MOV     C,L
        MVI     D,FBAS          ;BARE FISH
        LDA     FBA
        MOV     E,A
        CALL    FILL            ;LOAD SEA WITH FISHES
        LHLD    NOS
        MOV     B,H
        MOV     C,L
        MVI     D,SBAS          ;BARE SHARK
        LDA     SBA
        MOV     E,A
        CALL    FILL
        MVI     A,3
        STA     RCYC            ;SPEED-UP RANDOMIZER
        RET

FILL:   CALL    EMPT            ;FIND EMPTY RANDOM LOCATION
FLL1:   CALL    RAND
        ANI     7
        CMP     E
        JNC     FLL1
        ORA     D
        MOV     M,A
        DCX     B
        MOV     A,B
        ORA     C
        JNZ     FILL
        RET

CLR:    LXI     H,SEA
        LXI     B,SEAL
CLR1:   MVI     M,0
        INX     H
        DCX     B
        MOV     A,B
        ORA     C
        JNZ     CLR1
        RET

EMPT:   CALL    RAND
        MOV     L,A
        CALL    RAND
        ANI     7
        MOV     H,A
        CALL    INDX
        MOV     A,M
        ANA     A
        JNZ     EMPT
        RET

INDX:   PUSH    D
        LXI     D,SEA
        DAD     D
        POP     D
        RET

;       FOR ALL SEA, IF FISH MAKE RANDOM MOVE

FSCA:   LXI     H,SEA
        LXI     B,SEAL
FSC1:   MOV     A,M
        ANI     MASK
        CPI     FBAS
        CZ      FMOV            ;FOUND UNMOVED FISH
        INX     H
        DCX     B
        MOV     A,B
        ORA     C
        JNZ     FSC1
        RET
```

```
FMOV:   PUSH    B               ;SAVE COUNTER
        MVI     B,0             ;SEARCH CONDITION
        CALL    PICK            ;FIND MOVE THAT MEETS COND.
        JZ      FMV1            ;NONE SO DO NOTHING
        MOV     A,M             ;FETCH FISH
        INR     A               ;INCREASE AGE
        ANI     7               ;MASK AGE
        MOV     B,A
        LDA     FBA             ;BREEDING AGE
        CMP     B
        MVI     A,FBAS+MF       ;MOVED FISH
        JNZ     FMV2
        MOV     M,A             ;REJUVENATE FISH
        STAX    D               ;NEW FISH
FMV1:   POP     B
        RET

FMV2:   ORA     B               ;ADD AGE TO FISH
        STAX    D               ;RETURN TO SEA
        MVI     M,0             ;CLEAR OLD POSITION
        POP     B
        RET

DTBL:   DB      1,0,0,-1,-1,0,0,1
        DB      1,0,0,-1,-1,0,0,1
        DB      1,0,0,-1,-1,0,0,1

;       SEARCH FOUR ADJACENT LOCATIONS
;       B = SEARCH PATTERN, HL = CURRENT LOCATION
;       BC, DE LOST. HL UNCHANGED. DE = FOUND LOCATION
;       ZERO FLAG IF NO MATCH FOUND

PICK:   PUSH    H
        MOV     A,H
        SUI     SEA/256
        MOV     H,A
        MOV     E,L
        DAD     H
        DAD     H
        MOV     D,H             ;DE = UNMASKED XY
        CALL    RAND            ;A = RANDOM BYTE
        ANI     3               ;A = 0 TO 3
        ADD     A
        LXI     H,DTBL          ;DIRECTION TABLE
        CALL    ADHL            ;INDEX INTO TABLE
        MVI     C,4
PIC1:   PUSH    D               ;SAVE XY POSITION
        MOV     A,E
        ADD     M
        INX     H
        ANI     3FH
        MOV     E,A
        MOV     A,D
        ADD     M
        INX     H
        RRC
        RRC
        MOV     D,A
        ANI     0C0H
        ORA     E
        MOV     E,A
        MOV     A,D
        ANI     7
        ADI     SEA/256
        MOV     D,A             ;DE = NEW SEA ADDRESS
        LDAX    D               ;WHAT'S THERE?
        ANI     MASK            ;MASK NON-ESSENTIALS
        CMP     B
        JZ      PIC2
        POP     D               ;GET XY POSITION
        DCR     C               ;MORE DIRECTIONS?
        JNZ     PIC1
        POP     H               ;CURRENT ADDRESS
        RET                     ;NOTHING FOUND
PIC2:   POP     H               ;DISCARD XY
        POP     H               ;OLD ADDRESS
        MVI     A,1
        ANA     A               ;SET NOT ZERO
        RET

;       MOVE SHARKS

SSCA:   LXI     H,SEA
        LXI     B,SEAL
SSC1:   MOV     A,M
        ANI     0C0H
        CPI     SBAS            ;UNMOVED SHARK?
        CZ      SMOV            ;FIND SHARK MOVE
        INX     H
        DCX     B
        MOV     A,B
        ORA     C
        JNZ     SSC1
        RET

;       SELECT MOVE FOR SHARK
;       BC = CURRENT POSITION, HL = SEA ADDRESS

SMOV:   PUSH    B               ;SAVE COUNTER
        MVI     B,FBAS          ;MASK FOR FISH
        CALL    PICK
        JZ      NEAT            ;NO FISH
        MVI     C,0             ;STARVE LEVEL
        JMP     SAGE

NEAT:   MVI     B,0             ;MASK FOR SPACE
        CALL    PICK
        JZ      SMV1            ;NO SPACE
        MOV     A,M             ;FETCH SHARK
        ADI     8               ;STEP STARVE
        ANI     38H
        MOV     C,A             ;STARVE AGE
        LDA     SSA
        CMP     C
        JZ      DIE             ;STARVE
SAGE:   MOV     A,M             ;FETCH SHARK AGAIN
        INR     A               ;STEP AGE
        ANI     7
        MOV     B,A             ;AGE
        LDA     SBA             ;BREED AGE
        CMP     B
        MVI     A,SBAS+MF       ;MOVED SHARK
        JZ      SBRD            ;BREED
        ORA     B               ;AGE
        ORA     C               ;STARVE LEVEL
        STAX    D               ;LOAD NEW SHARK
DIE:    MVI     M,0             ;DELETE OLD SHARK
SMV1:   POP     B               ;RESTORE COUNTER
        RET

SBRD:   MOV     M,A             ;NEW BORN SHARK
        ORA     C               ;STARVE LEVEL
        STAX    D               ;OLD HUNGRY SHARK
        POP     B
        RET

;       DISPLAY SEA AND CONTENTS

DISP:   LXI     H,7000H
        LXI     D,SEA
DSP1:   LDAX    D
        ANI     7FH             ;CLEAR MOVED FLAG
        STAX    D
        ANI     MASK
        CPI     FBAS
        LXI     B,BLNK
        JC      DSP2
        LXI     B,SSYM
        JNZ     DSP2
        LXI     B,FSYM
DSP2:   MOV     M,C
        INX     H
        MOV     M,B
        INX     H
        INX     D
        MOV     A,H
        CPI     80H
        JNZ     DSP1
        RET

SCOR:   CALL    CENS            ;COUNT FISHES & SHARKS
        CALL    STR
        DB      8,6,3,'         ',8,0
        LHLD    NOF
        CALL    PRNT
        LHLD    NOS
        CALL    PRNT
        RET

CENS:   LXI     H,SEA
        LXI     B,0
        LXI     D,0
CEN1:   MOV     A,M
        ANI     MASK
        JZ      CEN2
        INX     B               ;FISH COUNT
        CPI     FBAS
        JZ      CEN2
        DCX     B               ;UNCOUNT FISH
        INX     D               ;SHARK COUNT
CEN2:   INX     H
        MOV     A,H
        CPI     ESEA/256
        JNZ     CEN1
        XCHG
        SHLD    NOS
        MOV     L,C
        MOV     H,B
        SHLD    NOF
        RET
```

```
;       PRINT FISH AND SHARK COUNTS

PRNT:   LXI     B,0
        MOV     E,C
        PUSH    B
        LXI     B,10
        PUSH    B
        LXI     B,100
        PUSH    B
        LXI     B,1000
PRT1:   PUSH    B
        MOV     A,B
        CMA
        MOV     B,A
        MOV     A,C
        CMA
        MOV     C,A
        INX     B
        MVI     A,-1
PRT2:   INR     A
        DAD     B
        JC      PRT2
        POP     B
        DAD     B
        ADI     '0'
        INR     E
        CPI     '0'
        JNZ     PNT3
        DCR     E
        JNZ     PNT3
        MVI     A,' '
PNT3:   CALL    TTO
        POP     B
        MOV     A,C
        ANA     A
        JNZ     PRT1
        MOV     A,L
        ADI     '0'
        CALL    TTO
        MVI     A,9
        CALL    TTO
        RET

;       RANDOM NUMBER GENERATOR

RAND:   PUSH    B
        PUSH    H
        LDA     RCYC
        MOV     C,A
        LHLD    RANN
RND1:   MOV     A,H
        DAD     H
        ANI     0A6H
        JPE     RND2
        INX     H
RND2:   DCR     C
        JNZ     RND1
```

```
        SHLD    RANN
        MOV     A,L
        POP     H
        POP     B
        RET

;       IN-LINE STRING PRINT

STR:    POP     H
        MOV     A,M
        INX     H
        PUSH    H
        ANA     A
        RZ
        CALL    TTO
        JMP     STR

        ORG     86E0H

STK:
IBUF:   DS      6
NOF:    DS      2
NOS:    DS      2
FBA:    DS      1
SBA:    DS      1
SSA:    DS      1
RANN:   DS      2
RCYC:   DS      1

        ORG     8700H

SEAL    EQU     800H

SEA:    DS      SEAL
ESEA:

FBAS    EQU     20H             ;BASIC FISH
SBAS    EQU     40H             ;BASIC SHARK
MASK    EQU     60H             ;IDENTIFICATION MASK
MF      EQU     80H             ;MOVED FLAG

BLNK    EQU     2020H           ;BLANK
SSYM    EQU     2173H           ;SHARK SYMBOL
FSYM    EQU     2670H           ;FISH SYMBOL

ESCD    EQU     16FFH           ;(16FFH)
CRLF    EQU     17C1H           ;(17C1H)
TTO     EQU     17C8H           ;(17C8H)
ADHL    EQU     194EH           ;(194EH)

UINP    EQU     81C5H
KBFL    EQU     81DFH
TSEC    EQU     81BAH
TMIN    EQU     81B9H
RDY     EQU     81FFH

END     WTOR
```

**NOTICE:** Intelligent Computer Systems, Huntsville

I am sorry to report that the Muellers have moved back to Germany and ceased their business activities in the United States. Before leaving, they discussed with J. Norris the possibility of COLORCUE becoming their CCII software representative in the U.S. Nothing further has been heard about such an arrangement as of this issue. This means that there is no legal source for CCII software in this country. We hope to hear further from Irmgard Mueller about this and will report any change in status in CHIP. In the meantime, please contact COLORCUE about your software needs just in case we might be able to give assistance.

# SEARCH

## A string search program for the 8000.

Bob Mendelson
27 Somerset Place
Murray Hill, NJ 07974

This program searches for a given string starting at a given address, both parameters set by the operator. It prints the start address of the found string location followed by the full string and all the characters after the string until a non-alphanumeric character is reached. It then prints the end address of the long string.

The nice feature of the program is that it prints out a long string while requiring the operator to enter only the first few search characters. For example, names and addresses can be found by typing only enough characters to identify the required string. Even if several locations contain the same characters the string printout makes it easy to select the desired string.

Here is how it works. The keyboard flags are first cleared. The program then asks for the start address of the search. If the operator knows the approximate location it will save a bit of time, but even running through the full 65535 address space takes only a few seconds. The program now asks for the key characters of the search string. When this data is entered the search begins. The character at the search start ad?The program then goes back to the start address and proceeds to check each character to be certain each is alphanumeric (that is, having an ASCII value between 20H and 7FH.) If it is, it will be printed. If not, the alphanumeric print will stop and the end address will be printed by the EDIT2 routine.

The STR3 routine asks the operator to chose among "Continue", "Start a new string", or "End operation" options. This program has proven so valuable that it is best stored on EPROM for immediate and easy use.

```
                                  ;STRING SEARCH
                                  ;
                                  ;BY R. MENDELSON, MODIFIED FOR COLORCUE
                                  ;VERSION 3-23-85

                              ;THIS ROUTINE SEARCHES FOR A GIVEN STRING STARTING
                              ;AT A GIVEN ADDRESS BOTH SET BY THE OPERATOR
                              ;IT PRINTS START ADDRESS OF THE FOUND STRING LOCATION
                              ;FOLLOWED BY THE STRING AND ALL CHARACTERS AFTER IT,
                              ;UNTIL A NON-ALPHA-NUMERIC IS REACHED.  IT THEN PRINTS
                              ;THE END ADDRESS OF LONG STRING.
                              ;

                               ;EQUATES, COMMON
                               ;------------------------
0103                           CI       EQU      0103H    ;CONSOLE INPUT
0109                           CO       EQU      0109H    ;CONSOLE OUTPUT
010F                           LO       EQU      010FH    ;LIST OUT
9FFF                           KEYBF1   EQU      09FFFH   ;KYBD READY FLAG
9FDF                           KEYBF2   EQU      09FDFH   ;KEYBOARD FLAG
012A                           OSTR     EQU      0012AH   ;STRING OUTPUT
0EE4                           EXHL     EQU      0EE4H    ;PRINT CONTENTS OF HL ON SCREEN
0133                           EXPR     EQU      0133H    ;INPUT 4 DIGIT HEX ADDDR
0006                           COLOR    EQU      06H      ;CCI STATUS
0010                           BLACK    EQU      16
0011                           RED      EQU      17
0013                           YEL      EQU      19
0016                           CYAN     EQU      22
000D                           CR       EQU      13
000A                           LF       EQU      10
000C                           EP       EQU      12
000E                           BA7ON    EQU      14
000F                           BA7OFF   EQU      15          ;BLINK A7 OFF
001D                           FORGND   EQU      29
001E                           BKGND    EQU      30
00EF                           EOS      EQU      239         ;END OF STRING

                                ;EQUATES, SEARCH
                                ;---------------------
9E20                            ADDR1    EQU      9E20H     ;START OF SEARCH ADDRESS
9E22                            LEN1     EQU      9E22H     ;LENGTH OF STRING
9E24                            ADDR2    EQU      9E24H     ;START OF 'SEARCH' ADDRESS
9E26                            ADDR3    EQU      9E26H     ;END OF STRING ADDRESS
9E30                            STR1     EQU      9E30H     ;STRING BEING SOUGHT

0000                                     .IFP1
                                         .PRINT  'SEARCH FOR STRING'
                                         .PRINT  'VERSION 3-85'
                                         .PRINT  'ENTER ROUTINE ORIGIN '
                                         .INPUT  RBEGIN
                                         ENDIF

0000                                     ORG     RBEGIN

F000 AF                         SEARCH:  XRA     A         ;ZERO
F001 32 FF 9F                            STA     KEYBF1    ;KYBD FLAG 1
F004 32 DF 9F                            STA     KEYBF2    ;KYBD FLAG 2

F007 CD 76 F1                   SERCH:   CALL    MESS      ;PRINT MESSAGE

F00A 06 13 0E                   ADDR:    DB      COLOR,YEL,BA7ON,BKGND,BLACK,FORGND,EP
     1E 10 1D
     0C
F011 53 45 41                            DB      'SEARCH-',BA7OFF
     52 43 48
     2D 0F
```

```
F019 0D 0A 0A          DB     CR,LF,LF,CYAN,'START ADDRESS > ',RED
     16 53 54
     41 52 54
     20 41 44
     44 52 45
     53 53 20
     3E 20 11
F02E EF                DB     EOS
F02F CD 61 F1          CALL   HEXIN  ;
F032 22 20 9E          SHLD   ADDR1  ;STORE ADDRESS OF SEARCH START

F035 21 68 F1  STR:    LXI    H,MSG1

F038 CD E4 F0          CALL   INPUT

F03B 2A 20 9E  BEGIN:  LHLD   ADDR1  ;LOAD STARTING ADDRESS
F03E 11 30 9E  CONT:   LXI    D,STR1 ;LOAD STRING
F041 3A 22 9E          LDA    LEN1   ;LOAD STRING LENGTH
F044 47                MOV    B,A    ;MOV LENGTH TO B

               ;COMPARE STRING WITH MEMORY

F045 1A        COMP:   LDAX   D      ;LOAD STRING BYTE
F046 BE                CMP    M      ;COMPARE WITH MEMORY
F047 C2 53 F0          JNZ    NOTEQ  ;BRANCH IF NOT EQUAL
F04A 23                INX    H      ;INCREMENT HL POINTER
F04B 05                DCR    B      ;DECREMENT LENGTH COUNTER
F04C CA 5D F0          JZ     EQUAL  ;BRANCH IF DONE
F04F 13                INX    D      ;INCREMENT STRING POINTER
F050 C3 45 F0          JMP    COMP   ;CONTINUE TESTING

               ;STRING DID NOT COMPARE, ADVANCE MEMORY START POINT

F053 2A 20 9E  NOTEQ:  LHLD   ADDR1  ;LOAD NEW START ADDRESS
F056 23                INX    H      ;INCREMENT
F057 22 20 9E          SHLD   ADDR1  ;SAVE END ADDR OF FIND
F05A C3 3E F0          JMP    CONT   ;START TEST AGAIN

               ;PATTERN DID COMPARE

F05D 3A 22 9E  EQUAL:  LDA    LEN1   ;POINT TO LENGTH
F060 2B        COUNT:  DCX    H      ;REDUCE ADDR AND COUNT
F061 3D                DCR    A
F062 C2 60 F0          JNZ    COUNT  ;ZERO?
F065 22 24 9E          SHLD   ADDR2  ;SAVE START ADDR OF FIND
F068 CD 76 F1  STOP:   CALL   MESS
F06B 0D 0A 0A          DB     CR,LF,LF,YEL,'ADDRESS',RED,' > '
     13 41 44
     44 52 45
     53 53 11
     20 3E 20
F07A 13 EF             DB     YEL,EOS
F07C 2A 24 9E          LHLD   ADDR2  ;RECOVER THE START ADDR TO HL REG
F07F CD E4 0E          CALL   EXHL   ;PRINT START ADDR OF STRING

F082 CD 76 F1  EDIT1:  CALL   MESS
F085 16 20 EF          DB     CYAN,' ',EOS
F088 2A 24 9E          LHLD   ADDR2  ;GET START ADDR OF FIND
F08B 7E        EDIT1A: MOV    A,M    ;GET STRING CHARACTER
F08C 23                INX    H

               ;IS CHAR NON-ALPHA-NUMERIC?

F08D D6 20             SUI    20H    ;IS IT LESS THAN 20H?
F08F DA 9D F0          JC     EDIT2  ;YES, END STRING PRINT

F092 D6 60             SUI    60H    ;IS IT GREATER THAN 7FH?
                                     ;-60H=7FH-1FH, 20H BECOMES 1FH BECAUSE-
                                     ;---SUBTRACT IS COMPLIMENT + 1
F094 D2 9D F0          JNC    EDIT2  ;YES, END STRING PRINT
F097 CD 09 01          CALL   CO
F09A C3 8B F0          JMP    EDIT1A ;LOOK AGAIN
F09D 22 26 9E  EDIT2:  SHLD   ADDR3  ;SAVE END ADDR OF STRING
F0A0 CD 76 F1          CALL   MESS
F0A3 13 20 54          DB     YEL,' TO',RED,' > ',YEL,EOS
     4F 11 20
     3E 20 13
     EF
F0AD 2A 26 9E          LHLD   ADDR3  ;RECOVER END ADDR (1 BYTE AFTER STR +EF)
F0B0 2B                DCX    H      ;ADJUST TO LAST CHARACTER ADDRESS
F0B1 2B                DCX    H      ;
F0B2 22 26 9E          SHLD   ADDR3  ;SAVE END ADDR OF STRING
F0B5 CD E4 0E          CALL   EXHL   ;PRINT ADDR OF END OF STRING
```

```
FØB8 CD 76 F1   STR3:   CALL    MESS    ;
FØBB ØD ØA ØA           DB      CR,LF,LF,RED,'C',CYAN,'ONT, '
     11 43 16
     4F 4E 54
     2C 20
FØC6 11 4E 16           DB      RED,'N',CYAN,'EW, '
     45 57 2C
     20
FØCD 11 45 16           DB      RED,'E',CYAN,'ND ) ',EOS
     4E 44 20
     3E 20 EF
FØD6 CD 03 01           CALL    CI      ;GET ANSWER
FØD9 FE 43              CPI     'C'     ;IF ANS IS Yes THEN
FØDB CA 3E FØ           JZ      CONT    ;CONTINUE TO LOOK FOR SAME STRING FROM
                                        ;END OF ADDRESS OF LAST FIND
FØDE FE 4E              CPI     'N'     ;IF ANS IS New THEN
FØEØ CA Ø7 FØ           JZ      SERCH   ;BEGIN AGAIN

                ;ANY OTHER CHARACTER WILL TERMINATE PROGRAM

FØE3 CF                 RST     1       ;TERMINATE PROGRAM

                ;INPUT ROUTINE
                ;-------------

                ;INPUT ROUTINE: ALLOWS BACKSPACE (ERASES CHARACTER),
                ;ERASE LINE AND ALSO LINE EDIT.  BACKSPACE TO THE ERROR,
                ;CORRECT AND CURSOR RIGHT TILL END OF INPUTS.  CURSOR
                ;WILL GO NO FURTHER.

FØE4 E5         INPUT:  PUSH    H       ;SAVE POINTER TO STRING
FØE5 CD 2A 01           CALL    OSTR
FØE8 21 30 9E           LXI     H,STR1  ;POINT TO OUR INPUT BUFFER
FØEB AF                 XRA     A       ;ZERO 'A'
FØEC 06 20              MVI     B,20H
FØEE 77         INPUT1: MOV     M,A
FØEF 23                 INX     H
FØFØ 05                 DCR     B
FØF1 C2 EE FØ           JNZ     INPUT1  ;ZERO 32 PLACES IN THE BUFFER

FØF4 21 30 9E   Y1:     LXI     H,STR1  ;POINT TO THE BUFFER
FØF7 06 00              MVI     B,Ø     ;'B' AS A COUNTER
FØF9 CD 03 01   Y2:     CALL    CI      ;INPUT 1 HIT OF KB
FØFC FE 1A              CPI     1AH     ;IS IT A BACKSPACE ?
FØFE CA 16 F1           JZ      Y4      ;IF YES, JUMP
F101 FE ØB              CPI     ØBH     ;IS IT AN ERASE LINE ?
F103 CA 2B F1           JZ      Y6      ;IF YES, JUMP
F106 FE 19              CPI     19H     ;IS IT A CURSOR RIGHT ?
F108 CA 50 F1           JZ      Y8      ;IF YES, JUMP
F10B FE ØD              CPI     ØDH     ;IS IT A 'RETURN' ?
F10D CA 38 F1           JZ      Y7      ;IF YES, JUMP
                ;NONE OF THE ABOVE

F110 77                 MOV     M,A     ;PUT INPUT IN BUFFER
F111 23         Y3:     INX     H       ;INCREMENT BUFFER
F112 04                 INR     B       ;INCREMENT COUNTER
F113 C3 F9 FØ           JMP     Y2      ;LOOP till ONE OF ABOVE

                ;COMES HERE IF INPUT IS A BACKSPACE

F116 2B         Y4:     DCX     H       ;DECREMENT BUFFER
F117 05                 DCR     B       ;DECREMENT COUNTER
F118 CA 2B F1           JZ      Y6      ;IF RESULT OF BACKSPACE => THEN..
F11B FA 2B F1           JM      Y6      ;INPUTS, ERASE LINE & DO ALL OVER
F11E 3E 20              MVI     A,20H   ;ELSE,.........................
F120 CD ØF 01           CALL    LO      ;.......ERASE THE CHARACTER......
F123 3E 1A      Y5:     MVI     A,1AH   ;......BACK UP THE CURSOR.......
F125 CD ØF 01           CALL    LO
F128 C3 F9 FØ           JMP     Y2      ;GO GET NEXT INPUT

                ;COMES HERE IF INPUT WAS AN ERASE LINE

F12B 3E ØB      Y6:     MVI     A,ØBH.
F12D CD ØF 01           CALL    LO      ;ERASE THE LINE
F130 E1                 POP     H       ;GET POINTER TO STRING
F131 E5                 PUSH    H       ;SAVE IT AGAIN
F132 CD 2A 01           CALL    OSTR    ;REPRINT THE STRING
F135 C3 F4 FØ           JMP     Y1      ;GO DO IT ALL OVER AGAIN
```

```
                        ;COMES HERE IF INPUT WAS A 'RETURN'
F138 AF           Y7:      XRA     A          ;TEST FOR Cr W/O INPUT
F139 B8                    CMP     B          ;IS COUNTER STILL ZERO ?
F13A CA 2B F1              JZ      Y6         ;IF YES, DO IT OVER
F13D 78           Y7A:     MOV     A,B        ;NEEDED TO COUNT STRING LENGTH
F13E 32 22 9E              STA     LEN1       ;FOR SEARCH PRG.
F141 3E EF                 MVI     A,239      ;--PUT AN 'EF' AT END OF-----
F143 77                    MOV     M,A        ;--INPUTS, MAKING A STRING OF IT
F144 3E 0D                 MVI     A,13       ;Cr
F146 CD 0F 01              CALL    LO         ;SEND
F149 3E 0A                 MVI     A,10       ;Lf
F14B CD 0F 01              CALL    LO         ;SEND
F14E E1                    POP     H          ;ELIMINATE POINTER TO STRING
F14F C9                    RET
                        ;COMES HERE IF INPUT WAS A CURSOR RIGHT (_)

F150 7E           Y8:      MOV     A,M        ;FETCH CHARACTER
F151 A7                    ANA     A          ;IS MEMORY A ZERO ?
F152 CA 23 F1              JZ      Y5         ;IF YES, IGNORE CUR.RIGHT
F155 3E 1A                 MVI     A,1AH      ;MEMORY not A ZERO
F157 CD 0F 01              CALL    LO         ;BACK CURSOR UP ONE
F15A 7E                    MOV     A,M        ;FETCH THE CHARACTER
F15B CD 0F 01              CALL    LO         ;PRINT IT
F15E C3 11 F1              JMP     Y3         ;GO GET NEXT INPUT

F161 0E 01        HEXIN:   MVI     C,1        ;COUNTS FOR FOUR BYTES OF ADDRESS
F163 CD 33 01              CALL    EXPR       ;---FETCH BYTES TO STACK,THEN---
F166 E1                    POP     H          ;--PLACE DATA IN HL REG AS START ADDR.--
F167 C9                    RET

F168 0D 0A 16     MSG1:    DB      CR,LF,CYAN,'STRING > ',RED
     53 54 52
     49 4E 47
     20 3E 20
     11
F175 EF                    DB      EOS


                        ;CALL FOR MESSAGE PRINT

F176 E1           MESS:    POP     H          ;FETCH POINTER TO MESSAGE
F177 CD 2A 01              CALL    OSTR       ;PRINT MESSAGE
F17A E9                    PCHL               ;RETURN TO PROGRAM

F17B              REND:

F17B                       .IFP1
                           .PRINT  'PROGRAM ORIGIN = ',RBEGIN
                           .PRINT  'PROGRAM SIZE   = ',REND-RBEGIN
                           .PRINT  'PROGRAM END    = ',REND
                           ENDIF

F17B F000                  END     SEARCH
```

## *A BUG IN FASBAS!* Peter Hiner

I regret to advise those of you who have the latest version (v12.24) of FASBAS that I have allowed a fatal bug to creep in. This bug was not present in earlier versions, so the corrective action outlined below should be applied only to v12.24.

```
ESC W                 ; to be sure of a clean start
ESC D                 ; to enter FCS
LOAD FASBAS.PRG       ; in response to FCS
ESC E                 ; to return to BASIC
POKE 39005,4          ; to eliminate the bug
POKE 39234,53         ; optional entry to change
                      ;   display header to v12.25
ESC D                 ; back to FCS
SAVE FASBAS.PRG;25 82A0 173F
                      ; save update to disk
```

Please check each entry carefully before hitting RET, and at the end, check the disk directory to be sure you have saved FASBAS.PRG;25 with the same values for size and load address as for v12.24.

```
Columns         SIZE      LBC       LADR       SADR
should read:    002F      3F        82A0       82A0
```

Credit for discovery of this bug goes to Doug Van Putte who presented me with a BASIC program that appeared to compile satisfactorily but caused the assembler (FBASM) to crash. I discovered the reason for this to be an error in the size of output buffer allocated within FASBAS. Running on a 32K machine, FASBAS would overflow the top of RAM if a medium to large BASIC program was being compiled. On a 16K machine not even a small BASIC program could be compiled! I am sorry for any inconvenience caused. This debugged release of FASBAS will now be officially called v12.25. □

# TRACE — A printing disassembler for the CCII.

Thomas Wulff
80 Bowen Road
Churchville, NY 14428

TRACE is a program that allows the user to observe the operation of a machine language program one instruction at a time. On each machine language instruction, TRACE causes the instruction to take place and prints the results of the instruction on the screen. The "results" include a printout of the status of all the 8080 registers, the stack contents, memory addresses, and the program counter. TRACE differs from other disassemblers in that the operations actually occur. Therefore, JMPs, and CALLs are properly executed. TRACE acts as though it were a microprocessor but it only simulates a microprocessor, in a sense. For example, if the code being traced places characters on the screen, they will not actually appear there, but the register and memory contents will be accurately simulated.

The operation of TRACE is more clearly seen in the accompanying printout.

TRACE requires a single disk, at least 16K RAM, and a printer connected to the MODEM port. To operate the program, a machine language auxiliary program, PR0042.PRG;1 is loaded into memory. If PR0042 is on the same disk as TRACE, it will automatically load before tracing begins. If, for some reason, PR0042 becomes altered during the run, it may be reloaded by inserting the program disk and entering 'RUN 4000' from Basic.

The program prompts the user for several replies at the start of the program. The user must, at this time, specify the following parameters:

1) the number of printer lines per page. The default is 80 lines/page, although most printer paper will require an entry of 66 lines or less. The program will generate a page command based on this number. The title, page number, and

```
PAGE  1
GAME.PRG (CHRIS ZERR)
START ADDRESS =  36864

          CMD               !SZAPC! A ! B ! C ! D ! E ! H ! L !DATA SP ADR!MEM  MADR
----------------------------+-----+---+---+---+---+---+---+---+-----------+---------
36864  34 SHLD        16698!     !   !   !   !   !   !   !   !           !18508
36867 172 XRA H            !00010!  9!   !   !   !   !   !   !           !
36868  55 STC              !00011!   !   !   !   !   !   !   !           !
36869  58 LDA          8328!00011! 69!   !   !   !   !   !   !           !
36872  55 STC              !00011!   !   !   !   !   !   !   !           !
----------------------------+-----+---+---+---+---+---+---+---+-----------+---------
36873  48 *** UDF <<       !     !   !   !   !   !   !   !   !           !
36874  50 STA         36410!     !   !   !   !   !   !   !   !           !
36877  32 *** UDF <<       !     !   !   !   !   !   !   !   !           !
36878  80 MOV D,B          !     !   !   !   ! 66!   !   !   !           !
36879  83 MOV D,E          !     !   !   !   ! 69!   !   !   !           !
----------------------------+-----+---+---+---+---+---+---+---+-----------+---------
36880  87 MOV D,A          !     !   !   !   ! 69!   !   !   !           !
36881   0 NOP              !     !   !   !   !   !   !   !   !           !
36882  77 MOV C,L          !     !   !   ! 76!   !   !   !   !           !
36883 144 SUB B            !00110!  3!   !   !   !   !   !   !           !
36884 202 JZ          21250!     !   !   !   !   !   !   !   !           !
----------------------------+-----+---+---+---+---+---+---+---+-----------+---------
36887  88 MOV E,B          !     !   !   !   !   ! 66!   !   !           !
36888 172 XRA H            !00010! 75!   !   !   !   !   !   !           !
36889 189 CMP L            !10011!   !   !   !   !   !   !   !           !
36890  40 *** UDF <<       !     !   !   !   !   !   !   !   !           !
36891  83 MOV D,E          !     !   !   !   ! 66!   !   !   !           !
----------------------------+-----+---+---+---+---+---+---+---+-----------+---------
36892  76 MOV C,H          !     !   !   ! 72!   !   !   !   !           !
36893  79 MOV C,A          !     !   !   ! 75!   !   !   !   !           !
36894  41 DAD H            !10010!   !   !   !   !   !144!152!           !
36895 164 AND H            !01110!  0!   !   !   !   !   !   !           !
36896  50 STA         13877!     !   !   !   !   !   !   !   !           !
----------------------------+-----+---+---+---+---+---+---+---+-----------+---------
36899 166 AND M            !01010!  0!   !   !   !   !   !   !           !
36900 189 CMP L            !00001!   !   !   !   !   !   !   !           !
36901  40 *** UDF <<       !     !   !   !   !   !   !   !   !           !
36902  83 MOV D,E          !     !   !   !   ! 66!   !   !   !           !
36903  72 MOV C,B          !     !   !   ! 66!   !   !   !   !           !
----------------------------+-----+---+---+---+---+---+---+---+-----------+---------
36904  73 MOV C,C          !     !   !   ! 66!   !   !   !   !           !
36905  41 DAD H            !00001!   !   !   !   !   ! 33! 48!           !
36906  58 LDA          9284!00001!130!   !   !   !   !   !   !           !
36909  40 *** UDF <<       !     !   !   !   !   !   !   !   !           !
36910  49 LXI SP , 10547   !     !   !   !   !   !   !   !   !52705 10547!
----------------------------+-----+---+---+---+---+---+---+---+-----------+---------
```

start address are printed as a header on each page.

2) the number of characters/line for printout. The default width is 80 characters.

3) the title of the program. Any text header may be entered here, and will be printed at the top of each printout page. The header must be restricted to the number of characters/line specified above.

4) the start address of the program. This is the address at which TRACE will begin disassembly. You must be certain that the address you supply is a logical starting address of a program or interrupt. The starting address is in decimal.

5) the printout destination, (S)creen or (P)rinter.

6) the printer Baud rate. You will enter the exact rate (ie: '2400', or '300'). TRACE does not recognize the familiar 1-7 entries here.

After this preliminary information is entered, the program will 'trace' until a HLT instruction is reached or the ATTEN/BREAK key is pressed. Since TRACE uses a machine language program, the registers and other processor parameters are not destroyed when the program is re-run, provided that PR0042 has not been reloaded in the meantime.

TRACE places PR0042.PRG immediately after the Basic portion of TRACE. If you alter the Basic portion, in terms of its byte count, and get an OM error, you do not have to reassemble PR0042. At the beginning of the Basic program you will see 'A0 = 165'. This is used to set the top of Basic RAM. It is also used in placing PR0042 in the appropriate memory area. Increasing 'A0' will provide more Basic memory space and also load PR0042 to a higher address. This load includes address changes in PR0042. TRACE also identifies any undefined functions but does not operate on them.

TRACE is available from the CHIP library in the form of seven files, including all source code and the assembled versions. The following printout demonstrates a typical TRACE output for the first few lines of code from Chris Zerr's GAME.PRG from the animation article in COLORCUE, VOL VI, No 5, p 12. □

## *"Pesticidal Programming"* Using ***IDA***'s Monitor.

W. S. Whilly

As promised (last year!) we will continue with our exploration of IDA, this time looking at the Interpret command and a few esoteric facilities of IDA.

IDA's monitor operates on a pseudo-8080 processor; that is, the register contents are simulated on the CRT for readout purposes. The best way to explore this marvelous instructional and debugging aid is with a short test program. Listing I has an assembler printout of the program we will use. Type it on your screen editor now and assemble it.

To explore a PRG file with IDA's monitor, first RUN IDAE (or IDA4). Next, I usually fill a good portion of memory with 00H so I can tell exactly where my program begins and ends. To do this type as follows:

```
IDA>F 8200 DFFF 00
```

Now we can load the PRG file from IDA:

```
IDA>XLDA MON.PRG
```

We do not specify a loading address, so IDA will use the loading address in the directory (8400H). You may verify the presence of the program by disassembling at 8400H:

```
IDA>D 8200 15+
```

...and there it is! We will now step through the program one instruction at a time, using the 'I' command. Clear the screen with the ERASE PAGE key. Now type in this command:

```
IDA>I8400 8400
```

In this command format the first 8400 is the address at which you want to begin executing. The second 8400 is a break point. Making the two numbers the same allows an examination of all the initial conditions without executing any of the program instructions. What you should see is this line on the CRT:

```
8400   210000   LXI     H,0000H    ; ???
 PC   B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8400 0000 0000 0000 00 0 0 0 0 0 C3 40D2 40DC C33E C532 3281
```

The top line of the display shows the next instruction to be interpreted, in the usual IDA disassembly format. On the next line, from left to right, the display shows the address in the program counter (8400), the BC, DE, and HL registers (all 00H), the accumulator, five flag bits, the current contents of the memory location held in HL (really the first byte of the operating system, since HL holds 0000H), the IDA stack pointer (40D2 in my computer, but your contents may differ depending on what you have done most recently with the computer and which version of IDA you are using), the current stack contents (SP+0) plus the last three stack entries.

Each instruction, in turn, is processed by IDA by pressing the BLUE color key (or CTL T). Press this key once now. The following line will be displayed:

```
8403   39         DAD      SP               ; 9
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8403 0000 0000 0000 00 0 0 0 0 0 C3 40D2 4CF8 C33E C532 3281
```

The program counter has advanced to 8403 because LXI H,xxxx is a three byte instruction. You will notice a change in the stack contents (at the star in the line above) because IDA is using the stack as it runs the monitor. We are in the process of assigning our own stack so it will reflect only our own work and not that of IDA herself. Press BLUE again and you will see the effect of DAD SP:

```
8404   22009A   SHLD     9A00H             ; "@Z
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8404 0000 0000 40D2 00 0 0 0 0 0 EA 40D2 4CF8 C33E C532 3281
```

The HL registers now hold the current stack pointer address (40D2 for me). You'll notice that M has changed, because the memory pointer has changed. M does not hold a byte shown in the first stack position on the screen, as we would expect, because IDA is deceiving us a bit about the 'real' contents of HL. However, we will change all of that when our own stack is clearly defined. When we press BLUE again, we will be saving the IDA stack pointer at address 9A00H. You could leave the interpreter and use IDA's Peek provision to examine that address, but we've done a clever thing. Press BLUE:

```
8407   31009A   LXI      SP,9A00H           ; 1@Z
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8407 0000 0000 40D2 00 0 0 0 0 0 EA 40D2 4CF8 C33E C532 3281
```

You'll see the contents of address 9A00H in the stack pointer address (SP + 0), which is right where we placed it! The next instruction, LXI,SP 9A00H will establish our new stack. Press BLUE and see the following:

```
840A   010201   LXI      B,0102H       ; ???
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
840A 0000 0000 40D2 00 0 0 0 0 0 EA 9A00 40D2 0000 0000 0000
```

The stack pointer is now 9A00H, and since we cleared memory in that area, all the stack positions are 0000H except for the 9A00H we recently placed there. This makes it easy to follow stack operations. Press BLUE:

```
840D   168F     MVI      D,8FH      ; VO   143
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
840D 0102 0000 40D2 00 0 0 0 0 0 EA 9A00 40D2 0000 0000 0000
```

The BC register pair has some new visitors. Press BLUE:

```
840F   1EFF     MVI      E,FFH      ; ^    255
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
840F 0102 8F00 40D2 00 0 0 0 0 0 EA 9A00 40D2 0000 0000 0000
```

So does D. Press BLUE:

```
8411   210000   LXI      H,0000H    ; ???
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8411 0102 8FFF 40D2 00 0 0 0 0 0 EA 9A00 40D2 0000 0000 0000
```

...and E. Press BLUE:

```
8414   80       ADD      B          ; @
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8414 0102 8FFF 0000 00 0 0 0 0 0 C3 9A00 40D2 0000 0000 0000
```

..as does HL. Notice that the old memory byte is back (C3). Press BLUE:

```
8415   F5       PUSH     PSW        ; u
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8415 0102 8FFF 0000 01 0 0 0 0 0 C3 9A00 40D2 0000 0000 0000
```

The accumulator now hold the sum of 00H and 01H (A = A + B). ADD puts the sum of A + B in A. Press BLUE:

```
8416   81       ADD      C          ; A
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8416 0102 8FFF 0000 01 0 0 0 0 0 C3 99FE 0102 40D2 0000 0000
```

We have PUSHed the accumulator and the flag register onto the stack (0102). '01' is the accumulator. What is the '02'? It's bit 1 of the flag register. Even though all the flags are zero at this point, remember that the flag register is eight bits wide but only five of these bits are used for flags. There are three bits unaccounted for here, and I suspect they are being used by the 8080 for something. If someone knows 'for what' please contact me at once! We know, at least, that bit 1 is set, because that's the only way to get '02' in the flag register. [1] Press BLUE:

```
8417   F5       PUSH     PSW        ; u
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8417 0102 8FFF 0000 03 0 1 0 0 0 C3 99FE 0102 40D2 0000 0000
```

We have added C to A (A = A + C). The Parity flag has been set. This happens when the number of binary '1's in the A register, following an operation, is an even number. The binary representation of 03H is '0000 0011'. ('0' is an even number also for purposes of parity.) Now watch the stack as we interpret the next instruction. The stack pointer will decrement, our old stack contents (0102) will be moved over to the right, and the new contents put in its former place on the CRT. Press BLUE:

```
8418   320090   STA      9000H      ; 2@P
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8418 0102 8FFF 0000 03 0 1 0 0 0 C3 99FC 0306 0102 40D2 0000
```

We have added to the stack. Notice that the PSW reflects the flag changes we made. Next with STA 9000H, we will save the A register at address 9000H. You won't see anything yet in IDA except the change in the program counter. You could always Peek at 9000H to be sure it's there. Press BLUE:

```
841B   82       ADD      D          ; B
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
841B 0102 8FFF 0000 03 0 1 0 0 0 C3 99FC 0306 0102 40D2 0000
```

Now we will add D to A. Press BLUE:

```
841C   83       ADD      E          ; C
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
841C 0102 8FFF 0000 92 0 0 1 0 1 C3 99FC 0306 0102 40D2 0000
```

A = 92H. The Sign flag has been set. This happens when bit 7 of A holds a binary '1'. 92H is such a number ('1001 0010'). We will now add '0FFH' to 92H. Press BLUE:

```
841D   F5       PUSH     PSW        ; u
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
841D 0102 8FFF 0000 91 1 0 1 0 1 C3 99FC 0306 0102 40D2 0000
```

What's this?! It seems 92H + 0FFH = 91H. Is IDA poor in arithmetic? You will notice the Carry flag has been set, indicating an A register overflow. The 'real' answer should be '191F' but a single eight-bit register can hold no more than 'FFH'. So the 8080 has done what you and I do with a column overflow in addition, it has generated a 'carry.' If your program were adding large numbers, some provision would have to be made for using this overflow to adjust the higher portions of a large number accordingly. We are going to PUSH this high number and its flags onto the stack. Press BLUE:

```
841E   EB          XCHG                 ; K
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
841E 0102 8FFF 0000 91 1 0 1 0 1 C3 99FA 9193 0306 0102 40D2
```

Next we will exchange the contents of the DE and HL register pairs. The numbers involved are not arbitrary for we are preparing to verify our STA 9000H instruction from line 13. Press BLUE:

```
841F   23          INX     H            ; #
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
841F 0102 0000 8FFF 91 1 0 1 0 1 00 99FA 9193 0306 0102 40D2
```

The next instruction will place 9000H in the HL registers. Notice that the contents under 'M' will change. Press BLUE:

```
8420   77          MOV     M,A          ; w
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8420 0102 0000 9000 91 1 0 1 0 1 03 99FA 9193 0306 0102 40D2
```

'M' shows the 03H we placed there way back in line 13. Let's copy our new A value to M. Press BLUE:

```
8421   97          SUB     A            ; W
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8421 0102 0000 9000 91 1 0 1 0 1 91 99FA 9193 0306 0102 40D2
```

Now we will subtract A from itself, which is a good way to clear the accumulator. Press BLUE:

```
LISTING I.    ;MON.PRG: Demo program for IDA's Interpret Command.

              ;NOTE: before loading this program, clear memory
              ;      from 8200 to DFFF.   [IDA>F 8200 DFFF 00]

  0000 (8400)           ORG     8400H

  8400 210000  BEGIN:   LXI     H,0000H         ;Clear HL registers
  8403 39               DAD     SP              ;Double-add current
                                                ; stack pointer to HL
  8404 22009A           SHLD    9A00H           ;Save stack address
                                                ; for end of program.
  8407 31009A           LXI     SP,9A00H        ;New stack pointer

  840A 010201           LXI     B,0102H         ;Set initial parameters
  840D 168F             MVI     D,8FH           ; with both MVI and LXI
  840F 1EFF             MVI     E,0FFH
  8411 210000           LXI     H,0000H         ;Clear HL

  8414 80      START:   ADD     B               ;A = A + B = 01H
  8415 F5               PUSH    PSW             ;Look at stack contents!
  8416 81               ADD     C               ;A = A + C = 03H
  8417 F5               PUSH    PSW
  8418 320090           STA     9000H           ;Place sum in memory
  841B 82               ADD     D
  841C 83               ADD     E
  841D F5               PUSH    PSW

  841E EB               XCHG                    ;Swap HL and DE
  841F 23               INX     H               ;Increment HL to 9000H, our
                                                ; memory location
  8420 77               MOV     M,A
  8421 97               SUB     A               ;Now A = 0, note zero flag
  8422 77               MOV     M,A             ;Zero M

  8423 D5               PUSH    D               ;Tricky way to get a lot
  8424 D5               PUSH    D               ; of zeros on the stack
  8425 D5               PUSH    D
  8426 F1               POP     PSW             ;Clear A and flag register
  8427 C1               POP     B               ;Clear BC

  8428 2A009A           LHLD    9A00H           ;Move contents of this
                                                ; location into HL
  842B F9               SPHL                    ;Now move it into SP
  842C 210000           LXI     H,0000H         ;Clear HL


  842F (8400)           END     BEGIN   ;Plus carriage return
```

```
8422   77       MOV    M,A       ; w
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8422 0102 0000 9000 00 0 1 1 1 0 91 99FA 9193 0306 0102 40D2
```

Notice that the Zero flag has been set. This happens when an operation causes the accumulator to go to 00H. We will clear the memory location by PUSHing this new value of A. Press BLUE:

```
8423   D5       PUSH   D         ; U
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8423 0102 0000 9000 00 0 1 1 1 0 00 99FA 9193 0306 0102 40D2
```

To finish up, we will clear a few more registers from the stack and restore IDA's stack. Press BLUE and watch this happen. When the first 'NOP' appears at program address 824F, you will be finished with this demonstration. Go ahead.

```
8424   D5       PUSH   D         ; U
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8424 0102 0000 9000 00 0 1 1 1 0 00 99F8 0000 9193 0306 0102
8425   D5       PUSH   D         ; U
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8425 0102 0000 9000 00 0 1 1 1 0 00 99F6 0000 0000 9193 0306
8426   F1       POP    PSW       ; q
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8426 0102 0000 9000 00 0 1 1 1 0 00 99F4 0000 0000 0000 9193
8427   C1       POP    B         ; A
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8427 0102 0000 9000 00 0 0 0 0 0 00 99F6 0000 0000 9193 0306
8428   2A009A   LHLD   9A00H     ; *@Z
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
8428 0000 0000 9000 00 0 0 0 0 0 00 99F8 0000 9193 0306 0102
842B   F9       SPHL             ; y
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
842B 0000 0000 40D2 00 0 0 0 0 0 EA 99F8 0000 9193 0306 0102
842C   210000   LXI    H,0000H   ; ???
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
842C 0000 0000 40D2 00 0 0 0 0 0 EA 40D2 4CF8 C33E C532 3281
842F   00       NOP              ; @
 PC  B C  D E  H L  A  C P A Z S M   SP  SP+0 SP+2 SP+4 SP+6
842F 0000 0000 0000 00 0 0 0 0 0 C3 40D2 4CF8 C33E C532 3281
```

IDA will (I)nterpret in a variety of ways. In the format Iaaaa bbbb, bbbb is a check point at which there will be a register dump. This check point may be in ROM or in user memory, and may be placed anywhere in the program. If bbbb is omitted, the program will run until its end or a previously set check point is reached. Such check points (up to eight of them) may be entered, one at a time with the 'C' command (ex: C840A). IDA will print all of these screen displays to the printer if you preceed the I command with an 'L' (ex: LI8400 8400). (This is how I obtained the copy for use in this article.)

I know of no better way to study the 8080 and its operation than writing simple programs (directly into IDA using the 'O' command if you wish) and watching them perform with 'I'.

Another of IDA's special features lies in the 'U' command. 'UA8200 8600' will display all the ASCII bytes in the memory range specified, 'UB8300 867A' all the data bytes, and 'UW8477 A09A' all the data words in the byte range. You will be pleased with the usefulness of this feature. See the IDA manual for more details.

More on restoring directories.

In my last article, we laborously discussed the restoration of a damaged disk directory. I have ruined many directories in my time, including the directory to the ISC SAMPLER disk just yesterday. I was away from home and doing some relatively urgent work. It was very, very embarassing! The Good Spirit reminded me that I had a duplicate of the disk on the reverse side so repair was very easy.

I placed the good disk in the drive and loaded IDA. I then read the directory plus some file bytes into memory:

```
IDA>XREA 00 8200 200
```

Now I put the bad disk in the drive and typed:

```
IDA>XWRI 00 8200 200
```

My directory was restored! This only works if the two disks are identical.

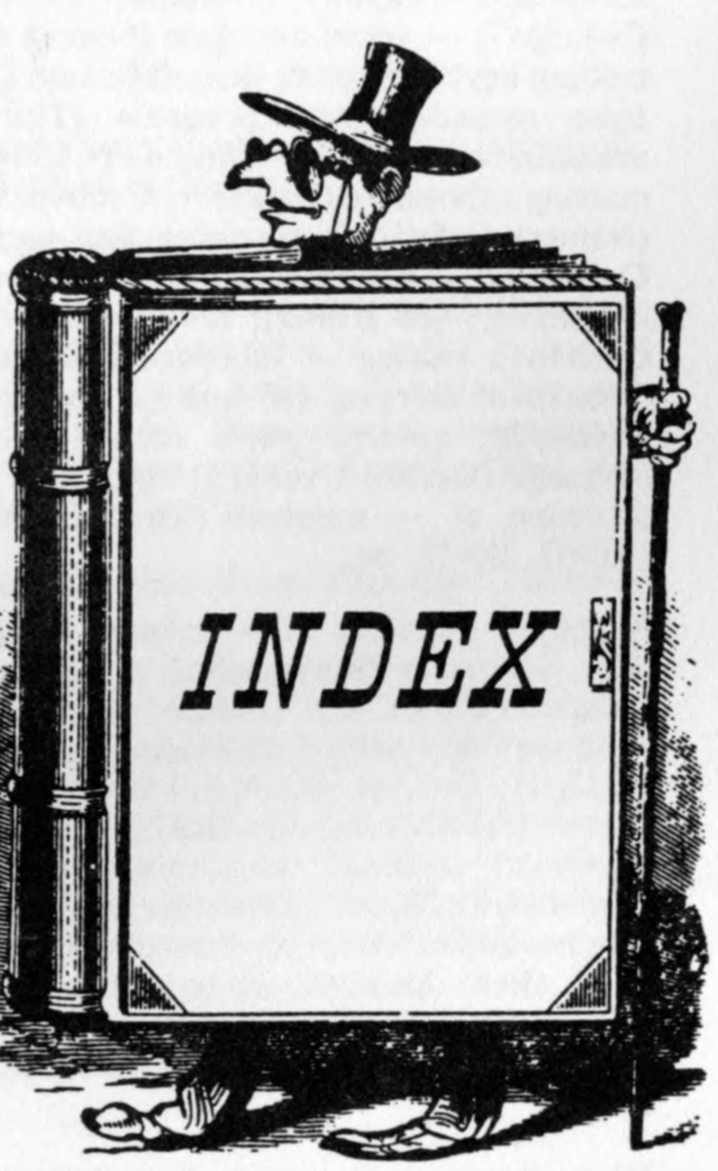
This completes our overview of IDA, the best software ever written for the CCII. We have not explored all the printing and editing facilities of IDA and I hope you will do this yourself. Hats off to Bill Greene. May he soon have IDA ready for CP/M and make some money!

[1] An interesting experiment for you. Try to determine the bit position of each of the flags by setting them in turn and examining the results of a PUSH PSW in IDA. The hex value shown on the stack will lead you to an exact determination of position for each of the flag bits. This chart shows the answers:

| BIT NUMBER | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| POSITION VALUE | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 |
| FLAG | C | x | P | x | A | x | Z | S |

*

Intecolor Corporation has its own bulletin board operated by George Price. It is called ***SPECTRUM*** and the telephone number is 404-446-6931. The board operates at 300—1200 baud.



# COLORCUE CompUKolour CUVIC DATA CHIP

*Prepared By Joseph Norris*

*The index in this final edition of COLORCUE covers published material from October 1978 through this issue of COLORCUE in August 1985 — eight years of user support. The following periodicals have been indexed in their respective time spans:*

*COLORCUE —October 1978 to August 1985, complete,*
*FORUM —March 1981 to the single issue of 1983, complete,*
*DATA CHIP —January 1979 to December 1984, complete,*
*CompUKolour —April 1982 to December 1984*
*CUVIC —January 1982 to June 1985.*

*It is not known if all the issues of the last two publications have been presented for indexing. If a few articles seem absent from the index I hope it will be because of this kind of omission.*

*Designing an index is somewhat of an art form. Realizing it, in any useful manner, is a most challenging occupation. As an intimate user of the CCII for five years I have relied heavily on my instincts to guide me in the selection of the index keys. 'Where would I look to find this?' has been my primary question as I read and catalogued the material. Unfortunately, my answer may not be yours in every case. I suggest that you scan the keywords on occasion to familiarize yourself with their pattern, and trust that this exercise will bear some fruit.*

*Most entries are made under more than one key. 'Printer', 'Handshake', and 'EPSON' are by their nature intertwined, for example, as are '50 pin bus' and 'interface.' All authors are listed by name with their articles following. Authors are also referenced in each entry, in parenthesis, where the authorship is known. (Some periodicals have been very careless about making authorship clear.)*

*Abbreviations are used as follows:*

*The first letter of the source for each listing refers to the publication, 'C' for COLORCUE, 'F' for FORUM, 'D' for DATA CHIP, 'K' for CompUKolour, and 'V' for CUVIC. This letter is followed in turn by the volume, number, and page, or a date of issue and page, as used by the several publications.*

*Several other abbreviations have been employed (somewhat inconsistently I fear) such as (A), meaning an assembly language program or routine, (B), meaning Basic code, (F), meaning Fortran, and so forth. 'Desc' means 'description.'*

*Page numbering is not always obvious in all periodicals. I have done the best I could under the circumstances. You may find several errors of a minor sort in this area.*

*The index is presented in a somewhat unusual fashion. FORUM and COLORCUE entries are grouped together in a single paragraph, as are CompUKolour and CUVIC. CHIP is in a paragraph by itself. This idiosyncracy is partially the result of a limited word processor capacity and the order of indexing, but also as a function of the likelihood of distribution of periodicals among readers in the United States and in other countries. At any rate, if you have only one of the periodicals, you may see, readily, what is available to you in your own library without reading the entire key listing.*

*There has been a considerable amount of reprinting and borrowing of articles amoung these publications. When this has been clear, duplicate entries for the same article will appear.*

*While COLORCUE will cease to exist with this issue, your access to the literature from which this index was derived will continue. If any subscriber wishes reprints of any article contained in this index, I will be pleased to provide them in XEROX form. The cost for each mailing will be $2.50 for the United States and Canada, and $5.00 in US funds for mailing outside the North American continent. Several articles may be included for the designated fee.*

*I send my thanks to the editors and subscribers among all the participants for their generousity and cooperation in this project. My special thanks to Wallace Rust who took valuable time to comment on an early version of the index and whose suggestions have improved it in great measure. I relieve him, however, from the burden of my errors which number, I am sure, as an early population of fishes in Tom Napier's 'watery world.'*

*Finally, I must add that I have been somewhat in awe of the monumental work performed by our prolific authors, whose identity becomes very apparent with the reading of the index. Their support and untiring dedication to us seems to me an extraordinary thing in this very commercial world. With their help we have been able to share in the joy and richness of the computer experience, and we have had the opportunity to sustain friendships of a high order over the years. May you find the index useful as you continue your exploration of the Compucolor II.*

**CUVIC.** — disk library listing Dn38p3

Subscriber list Oct, 1984 VOct84p8; disk library catalog (Jan85) VJan85p8, (Jul85) VJun85p9, (for 3651) VFeb85p2

**CUWEST.** Western Australia User Group Newsletter. Cv6n3p31 Cv6n3p31

**CYPHER.** Software demo (Whilly) Cv6n2p19

## D

**DAA.** 8080 instruction, see [assembly language]

**data base.** Assembly language — (Helms) desc Fv1n2p11, review Fv2n1p73

Discussion of . construction with some examples (Barlow?) Dn7p2; also see [Personal Data Base]

**DATA CHIP.** First issue January 1979, continuing to present. Editor: Ben Barlow. Subscriber list not published. Disk library: see [software]. The publication of the Compucolor II User Group of Rochester (N.Y.)

**date.** — stamping when duplicating files (Barlow?) Dn6p4

**Davis, Glen M.** "How not to be out of sorts." Sorting algorithms Fv1n2p24; "The old shell game." (B) programs for sorting, Fv1n4p20

**debugger.** "NEWBUG" Comtronics, review (Steffy) Fv2n4p3; "IDA" Bill Greene, review (Norris) Cv6n3p12; "TRACE" (Wulff) Cv6n6p33; also see [MLDP], [Suits], [Whilly]

Tutorial on debugging (Muldowney) VApr85p2

**decimal.** — to hex conversion (B) (Andries) Fv2n2p5

— to hex conversion routine (B) (Rusch) Dn12p5; — to binary conversion (B) Dn26p22; also see [conversion]

**DEL.** Explanation of delete process in FCS, Dn13p5

**DeVito, Mike.** "A programming quickie" graphics character generator Dn37p1

**Devlin, Jane.** "In and out of the Compucolor II" Cv4n5p13

**Devlin, Tom.** "Disk alignment for the Compucolor" instructions, Fv3n1p3; "Lower case y's" Cv4n1p13; "8K ram board for the Compucolor II" Cv4n5p5; "Program to 'reorg' FREDI" Cv4n5p11; "Screen memory problems and cures" Cv5n5p17; "Notes on the CRT controller chip" Cv6n4p12; "Merging Basic and Assembly language programs" Cv6n4p26 ref; "No-echo patch for Basic without assembly language" Cv6n5p31

"Lower case y's" Dn27p8

**Dewey, Dale.** "Compucolor II and the multifunction input/output controller." (TMS5501) + bibliography, Fv1n5p82; "Devlin vrs Orford cheap 'y' fix." Fv2n1p23; "Advanced programmer's manual." Private publication, $15. Describes ROM routines and flags. CHIP library, ref Fv1n2p15

"Do you have a bug in your Battleship program?" Dn18p10; considerations in selection of ROM chips for CCII Dn19p9; "Version compatibility routine" (jump routine for v6.78 & v8.79) (A) Dn19p9; "Video alignment for the CCII" Dn26p2; "The Last Bug" (poem) Dn28p2; FCS ROM bugs and corrections (listing) Dn33p13

**Diablo.** Software handshake for — 630 (Pinter) Cv5n2p11

**DIGITALKER.** (Rhijn) review (Holt) Fv2n1p60

**Dinsmore, Gary.** Stiffening CPU RESET against accidental press Cv5n2p4; "How Basic stores variables" Cv6n1p16; "User supported software" Cv6n2p3

**directory.** Data — (Durant) special use in industry described, Cv3n2p4; "What to do after you hit DIR." Explanation of directory display Cv2n5p11; — library, software (Power) Fv2n1p40; list — on clear screen, Fv1n1p9; dissecting a — (Martin) Cv3n5p9; restoring - (Steffy) Cv5n2p9; see also [disk drive]; reconstructing a - (Whilly) Cv6n4p16, (Mendelson) (8000) Cv6n4p22

— management software (B) (Rusch) Dn12p4

Basics of — (Donkin) Kn1p3; menu for — loading (B) (Donkin) Kn1p19; — edit program (B) (unknown) Kn2p32; DOCDIR program to edit directory (A) (Pankhurst) VOct82p6; recovering lost programs (Farquhar) VJan83p5; — and bugs in FCS ROM (Woods) VMay84p3

**DIRMOV.** Disk management program (B) (Rusch) Dn12p4

**disassembler.** See [debugger]

**disk.** — drive, maintenance (Gould) Fv2n3p18; adding 3rd — to CCII (Newman) Fv2n4p53; — alignment (Devlin) instructions for, Fv3n1p3; 8" drive from Netherlands, note, Fv1n5p33, update Fv2n1p4, Fv2n2p19; preventing computer turnoff with disks in drives (Green) Fv2n1p29; changing — (Pinter) Cv5n3p3; first aid for — (Herold) Cv5n3p10; — duplication, see [DUP], [Johnson]; — improvements (Newman) Cv5n4p20, update Cv5n5p18, Cv5n6p9; — utilities (B) (Napier) Cv5n5p12; replacing belts on — Cv6n5p25; — error, ESKF — (Richardson) Fv2n1p26; hard — for the CCII (Newby) desc and instructions Cv6n6p6; comment on the hard — drive (Zerr) Cv6n6p12; also see [repairs]

— load error, see [EDCS]; display — contents on CRT (B) (Johnson) Dn13p7; floppy — theory of operation (Kahkonen) Dn23p29; CCII — compatibility (unknown) Dn30p25; — alignment (Donkin) Dn31p7

Alignment of — drive (Donkin) Kn1p14; installation of LED in — drive (Winder) VOct82p8; correcting — drive problems (Winder) VNov82p4; conversion of foreign — drives to CCII (Allen) VMay84p5 also see [Farquhar]

**display.** — memory (software) (B) (Barlow) Dn12p10; — disk contents on CRT (B) (Johnson) Dn13p7; clock — routine (A) (Taylor) Dn20p14; also see [screen]

**DOCDIR.** Software by D. Pankhurst VOct82p6

**Donkin, Adrian.** "The Black Arts', notes on disk drive alignment, Kn1p14 Dn31p7

**Donkin, Bill.** "The Compucolor directory and Basic programmes on disc." Kn1p3; "Monopoly" notes on CHIP game Kn2p18

**downloading.** See [conversion], [APPLE], etc.

**DRAW.** Program to test graphic shapes (B) (unknown) Kn3p9

**drawing.** — on the CRT (B) (Barlow?) Dn7p4; also see [screen]

**driver.** See [printer]

**drives.** See [disk drive]

**DSP.** Graphics dump of screen (B) for Microline 80 (Stuckey) Fv2n2p14

**DSR.** Implementation of — for CCII (Wulff) Dn35p7

**dump.** Screen —, see [screen], [printer], or printer type.

**DUMP.** Mod to — program in CUVIC library (Lewis) VJan83p8

**DUNGEONS & DRAGONS.** Review (Stuckey) VAug83p5

**DUP.** Program, instructions, listing Cv2n1p5, correction Cv2n3p5; bugs in — (B) (Barlow) Dn3p5; also see [Johnson]

**duplication.** Creating time and date stamping for files while duplicating (Barlow?) Dn6p4; — of screen displays, see [screen], [Suits], [Steffy]

Disk duplication software (B) (Johnson) Dn1p1; — of screen displays with one drive (Suits) Dn8p5

**Durant, Judy.** "Data directory." Cv3n2p4;

**dust cover.** Plans for making a —, Cv2n4p2

**'Dynamic'.** — ellipse doodler, see [Napier]

## E

**editor.** Text — (B) program (Hogan) Fv1n4p26; also see [SCRIPT], [Steffy] , [Compuwriter], [CTE]

Screen — for FORTH, see [Greene] , [Napier]

**EDCS.** Overcoming — errors (Johnson) Cv3n6p20

Recovering from — file load error (Johnson) Dn12p3

**encryption.** See [CYPHER], [Whilly]

**Epps, Garry.** See ['THE' Word Processor] (review)

**EPROM.** — programmer (B) (Pankhurst) VOct83p11

See [bank selector], [character], [16K addon]

**Epson.** Screen dump to — with Graftrax (Rex) Fv2n4p36; commentary on —, cable connections, screen dump (Rex) Fv1n4p13; commentary on changes in — printers (Peel) Fv1n5p15; technical user commentary (Richardson) Fv2n1p30; graphics dump for Microline 80 (Stuckey) Fv2n2p14; screen dump to MX80 (Fairbrother) (B) Cv4n1p14, (A) (Reddoch) Cv4n6p9; see also ['y' fix], [printer]; note on GRAFTRAX 80 (Barlow) Cv4n2p4; MX80 graphics (Lowe) Cv6n2p22

Interfacing the — MX80 to the CCII (Fairbrother) Dn23p3; also see [Thompson, Paul]

Bug in — Microline 80 (Winder) VSep82p6; using MX80 with WORDKING (Stuckey) VDec82p8; working with the — printer to simulate a typewriter (Farquhar) VOct84p5

**ESC.** — vectors. See [Taubold] save/load screen display; adding software — vectors to v6.78 (Steffy) Fv2n3p42; -W, — E (Basic interpreter, Linden) Fv2n4p5

Recovery from — W (Montemarano) Dn26p28

**ESKF.** — disk error (Richardson) Fv2n1p26

**EXECUGRAPH.** Operation of software VAug84p7

**EXECUTE.** — command, 3651, operation (Hennig) Fv1n2p20

## F

**factoring.** — numbers (B) Cv2n5p5

— numbers (B) routine (Barlow) Dn4p5

**Faflick, Philip.** "The real apple of his eye." Reprint from TIME, Dn32p23

**Fairbrother, Mark.** "Screen dump to the MX80 printer" Cv4n1p14; "Sphere program" (graphics) (B) Cv4n2p15

"Epson MX80" review Dn23p3

**Farquhar, Jim.** "A layman's guide to recovering lost programs from disk." VJan83p5; "A suggestion for universal printer routines.' VJun83p9; "Program development in an entirely unstructured manner." VOct84p5

**FASBAS.** Basic compiler by Hiner, review (Suits) Cv5n4p26, update Cv5n5p3; — instruction manual Fv3n1p9 also see [Hiner]

"Fun with Basic compilation" (unknown) Kn4p7, (Booth) VOct83p4; notes on — (Winder) VJun85p2, also see [Hiner]

**FCS.** List volume name only, Fv1n1p9; changing default drive with POKE v6.78, Fv1n1p9; transferring file names accurately (v6.78) Fv1n1p9; list directory on clear screen, Fv1v1p9; executing - commands in Assembly (unknown) Cv3n5p13; overcoming EDCS errors (Johnson) Cv3n6p20; also see [COPY] and other FCS errors and commands; scratchpad RAM locations (flags, etc) Cv3n7p10 (Dec1980/Jan1981); — disk operations (A) see [Norris]; return to — (A) (Napier) Cv6n2p4, (Zerr) Cv6n5p23

**Rosen, Howard.** "Subroutine to initialize T-matrix and S1-matrix" (F) (software) Dn32p7

"Fortran programming." Reprint from Colorcue VDec82p7

**RS232.** Connecting to IBM PC using —, Fv2n2p18; — interface, baud rate Cv1n2p6; — interface (Barlow) Cv4n1p5; serial to parallel interface (Barlow) Cv4n3p13; using — to connect two Compucolors Fv2n2p18; also see [communication]

Using — for communications with other computers, Dn10p6; using — to transfer files between CCIIs (Rusch) Dn18p5; definitions and null modem connections (general discussion) Dn28p14; DSR implementation of — for CCII (Wulff) Dn35p7

Use of interrupts with — bus (Green) Kn2p27

**Rubik's cube.** See [Safford]

**RUN.** FCS — command, description (Minor) Dn29p11

**Rusch, Gordon.** (Com-Tronics) "Directory management" (DIRMOV software (B)) Dn12p7; "Down-loading documentation", exchanging files between CCIIs, Dn18p5

**Rust, Wallace.** "Precision in Basic numbers" Cv6n5p30; "CCII color adjustment" Cv6n5p32; "ASTRO" (software) (B) Cv6n6p18; "ASTRO" software Cv6n6p18

"Color show" (B) demo of CCII colors Dn2p5; "Creating displayed remarks in program listings" (B) Dn6p; "Simplified screen poking" Dn8p9; "ASKME, Artificial Intelligence program" desc Dn11p5; "Do you have a bug in your Air Raid program?" Dn18p10; chart of graphics characters, Dn25p7

## S

**Safford, Roger.** "Rubik's cube demystified" (B) Cv4n6p5

"All you really want to know about RND(X) but didn't care to ask" Dn2p1; note on extending random number sequence Dn10p4

**SAMPLER.** Modification to (Shanks) Cv2n7p19

**SAVE.** FCS — command, explained (Minor) Dn29p6

**SCAR.** Single character input routine (B) (Minor) Dn32p3

**Schefe, Neville.** "Computerised star map." VJan84p3

**screen.** Saving — displays (Taubold) Fv2n3p9; — editor (Comtronics) review (Steffy) Fv2n3p42, update (Steffy) Fv2n4p2; — editor, see [ISC]; — dump to Epson (Rex) (A) Fv2n4p36, (B) (Stuckey) Fv2n2p14; getting the best from your — display, alignment (Newman) Fv2n1p19; drawing on the — (A) (Manazir) Cv3n2p19; photographing the CCII — display (Grogono) Cv2n6p2; saving the — display (Steffy) (B) Cv2n5p13; — memory (Linden) Fv1n2p33; changing — displays (Suits) Fv1n3p35; — 'saver' (Hudson) moving characters non-destructively (B) Cv3n5p3; loading screen display by assembly language CALL from (b) (Steffy) Cv3n7p8 (Dec1980/Jan1981); — memory problems and cures (Devlin) Cv5n5p17; — alignment (Rosen) Cv5n5p19; — editor in FORTH (Napier) Cv6n5p8; — display addresses, character memory locations, large chars, alternate char sets, CCI (Linden) Fv1n2p36; 'Quick change artistry' (Suits) Fv1n3p35; — editor for small keyboards (Linden) Fv1n1p9; — memory (Linden) Fv1n2p33;

Copying — display to disk, software (B) (Johnson) Dn5p2; drawing on the — (B) (Barlow?) Dv7p4; — display utility software (B) (Suits) Dn8p5; — memory address table (Rust) Dn8p9; changing — displays rapidly, "Quick change artistry" (A) (Suits) Dn16p7; — refresh memory, explanation of duplicate addresses, Dn17p3; determining screen character with PEEK, simulation of TRS80 IF POINT statement (van Putte) Dn19p15; — alignment (Dewey) Dn26p2; — editor with MX80 (Thompson) Dn30p17

— editor by Doug Pankhurst (review) VSep82p5

**Scribe.** (Word processor) see [Steffy] SCRIPT;

**SCRIPT.** see [Steffy]

**scrolling.** "How to use the scrolling patch" (B) Cv1n1p2, correction Cv2n3p5; — patch in (B) (Linden) Fv1n4p4, analysis of — patch, begins Fv1n5p52, see [Taubold 'Ram batterings']

Corrections to — patch article in Cv1n1 (Suits) Dn10p5

**search.** — string software (Mendelson) Cv6n6p28

**serial.** — to parallel converter, review (Norris) Cv6n2p24; — port, see [RS232], [Barlow]

**Shanks, Bill.** Modification to SAMPLER (ISC) Cv2n7p19

**Shell-Metzner.** See [sort]

**SHOOT.** Software modification (Muldowney) Cv3n6p19

Program variations to — (Suits) Dn3p2

**shuffling.** — in (B) (Woods) Fv1n3p27; also see [cards] Card — (Taubold) Dn26p23

**Smith, Bob.** "Incremental plot table" Cv4n2p17; "CRT mode plotting" Cv4n4p17

**Smith, Graeme.** "Binary to ASCII/ ASCII to binary" Cv3n3p16

**Smith, Ken.** "A summary of some North Star Basic differences.' VSep82p4; "A simple index search program." VJan83p6; "Dictionary of Useless/Useful routines." (B) VMar83p4

**Smithy's Bulletin Board.** Instructions for access (Australia) VMay84p5, VMay84p4

**software.** Catalog of commercial — for CCII with some reviews (Norris) (May 1984) Cv6n3p26; additional sources Cv6n3p30; some software references under their name (such as [IDA]) Library holdings CHIP Dn38p6, recent additions Dn38p15: CUVIC VJun85, holdings in CHIP library Dn38p3: FORUM (source) see [FORUM]; CompUKolour Kn7p24; the following are reviews or references from FORUM (caps omitted for clarity):

Alien Invasion (review) Fv1n3p10; Assembly language data base (Helms) review Fv2n1p73; CHOMP (Pacman) review (Peel) Fv2n1p42; Colorcalc & Colorgraph (review) Mueller Fv1n5p38; Compucon Ltd Fv1n1p16; Foolsmate (chess) Fv1n5p13; Galaxian (review) Fv1n3p19; Invaders (review) Fv1n2p22; Super monitor (Greene) desc Fv1n2p28; 'THE' Basic Editor (early review) Fv1n1p22, (Perrigo) Fv1n4p24, update Fv2n1p11; Tracer, tracing a Basic program, desc Fv1n5p11; Trigonometry, Metra Instr Fv1n3p6

**sort.** Bubble —, alphabetical, numerical (Davis) Fv1n2p24, and Fv1n4p20; see [Davis] for general algorithms; Callable — routine (A) (Matzger) Cv4n2p21

— routine (B) (Barlow?) Dn7p2, (B) (Jenkins) Dn27p4

**sound.** Interface for — production (Barlow) Dn16p3; also see [Chamberlin]

**Soundware.** Commentary on kit (Winder) Fv1n5p31, Fv2n1p27; review (Peel) Fv2n1p28; — from MicroSynth (Mueller) review Fv2n2p27; work on — (Power) Fv2n1p40; see [Grant], [synthesizer]

— mod to CCII (Pankhurst) VFeb82p7; commentary (Standen) VFeb84p2

**South, N.** "Generalised inverse of a matrix." (B) VNov82p9

**SPECTRUM.** Program listing (Martin) Cv2n3p3

**speed.** Increasing — of (B) programs (Taubold) Dn29p15

**Spencer, John.** Tips on using FOR-NEXT loops VApr82p3; 'Programming finesse." VNov82p7

**Spracklen, Kathy.** Letter to D. Peel, re chess Fv1n3p9

**Squailia, Richard.** "Data base management —update control" (ISC personal database mod) Dn14p5

**'squeeze'.** See [Taubold]

**Stark, Aub.** "Understanding 'The Australian Beginning'." VDec82p4

**Standen, Peter.** "Bell for v6.78 Compucolor." VApr83p3; commentary on Soundware VFeb84p2

**Star Trek.** — strategy Cv1n1p6;

Changing quadrant colors in —, Dn7p2

**stack.** "Stuffed stack syndrome" (Steffy) Fv1n3p25

**Steffy, Myron.** "CTA assembler" review Fv2n3p17, update Fv2n4p3; "Assembly language subroutines" ROM listings, jump tables, keyboard character routine Fv1n5p44: baud, order of entry, numerical conversions, moving characters Fv2n1p13: justify, print and store text files, Fv2n3p24: SCRIPT handler program Fv2n4p29: "A 'jumper' for additional escapes" software, Fv2n3p42; "CTE screen editor" review Fv2n3p40, update Fv2n4p2; "Screen drawing with a joystick" (A) Fv2n3p46; "Newbug" review Fv2n4p3; "Assembly language subroutines" keyboard input of hex characters, Fv3n1p29; saving a screen display (B) Cv2n5p13; "Basic program Restoration" Fv1n3p23; "The stuffed stack syndrome" Fv1n3p25; "Tracking Basic variables" Cv3n3p24; "The CALL function" Cv3n7p8 (Dec1980/Jan1981); "Disk data recovery" Cv5n2p9; "A program to load SRC files into Compuwriter" Cv6n1p8; appreciation (Norris) Cv6n1p11

"A jumper for additional escapes" (A) Dn31p2, additions Dn32p15; "A method of splitting a program between two RAM cards" (A) Dn33p8

**string.** Search — program for 8000 (Mendelson) Cv6n6p28

**Stroop.** — phenomenon, Cv2n2p2;

**Stuckey, Peter.** "Microline 80 graphics dump" (B) Fv2n2p14

Header for Basic programs VMay82p1; "Printing for CUVIC with the Epson MX80." using WORDKING word processor VDec82p8; "How to do great things with soft apples and trash." Program conversion from other Basics VJnu83p11; "Disc reviews." (Dungeons & Dragons, Castlequest) VAug83p5

**style.** Notes on Basic speed and style (Norris) Cv5n3p11

Notes on programming—(from Kernighan) Kn7p11; "Good programming techniques" (unknown) VJun82p3; "Programming finesse." (Spencer) VNov82p7; notes on Basic—(Muldowney) VJun83p7

**subroutines.** (A) see [Steffy], [Matzger], [Norris], [Suits]; decimal to hex conversion (B) see [Andries]

**subscribers.** — to FORUM, list Fv3n1p34, to COLORCUE, list Cv6n5p26

**Suits, David.** "Reentering plot submodes." Cv3n2p11; "Color graphics for Intecolor.." review (Peel) Fv1n3p17; "Quick change artistry" Fv1n3p35; excerpt from "Color Graphics" (hatch character) Cv3n3p25; "Assembly language programming", Part 1 Cv4n1p19 registers, binary & hex numbers: Part 2 Cv4n2p6 using MLDP simple program in assembly: Part 3 Cv4n3p19 8080 instruction set, program topology: Part 4 Cv4n4p19 status flags and stack: Part 5 Cv4n5p19 the input routine: Part 6 Cv4n6p20 input routine cont: Part 7 Cv5n1p17 macro-assembler: Part 8 Cv5n2p24 "Simple math": Part 9 Cv5n3p5 numerical I/O and random numbers; "'THE' Basic editor" (review) Cv4n3p25; TECH TIP (space bar pressure) Cv5n1p15; 'How did Sam die?' and 'Was Einstein correct?' (CUTIES) Cv5n2p10; "Calendar printer" (B) Cv5n2p17; "Blue sky dept" enhancing CCII color capabilities Cv5n5p20; "Transformers (not electrical)" Cv5n6p17; book reviews Cv6n1p12; "One dimensional cellular automata" Cv6n3p16

"Variations for 'SHOOT'", Dn3p2; "Some comments on the Basic Utilities Disk" Dn8p2; "DISPLAY/CREATE/EDIT/DUP" discussion with software Dn8p5; "Tidbit #123" using large characters Dn8p10; "The scrolling patch" commentary and corrections to Colorcue article V1n1, Dn10p5; "Plotting character strings" (PLOT 2) Dn11p3; "An animated joke" (software) (B) Dn13p2; "Quick change artistry", changing screen displays rapidly, Dn16p7; "Some ideas and a quiz" Dn20p5; "The 'last' key code" realtime keyboard entry (B) Dn22p4; "The 'new' key code" Dn23p17; "ALPOCII", introduction to assembly language programming, Dn24p14, Dn26p17; "Instructions for 'Capture the flag'." Dn37p6

"Capture the flag." Software instructions VMar85p2

## INDEX: Additions, Corrections & Notes.

COLORCUE 19 West Second Street • Moorestown, NJ 08057